51CTO.com
技术成就梦想
我 们 只 谈 开 发

开发月刊

Development Monthly

2013年09月
总第030期

面霸的八月：小米面试记

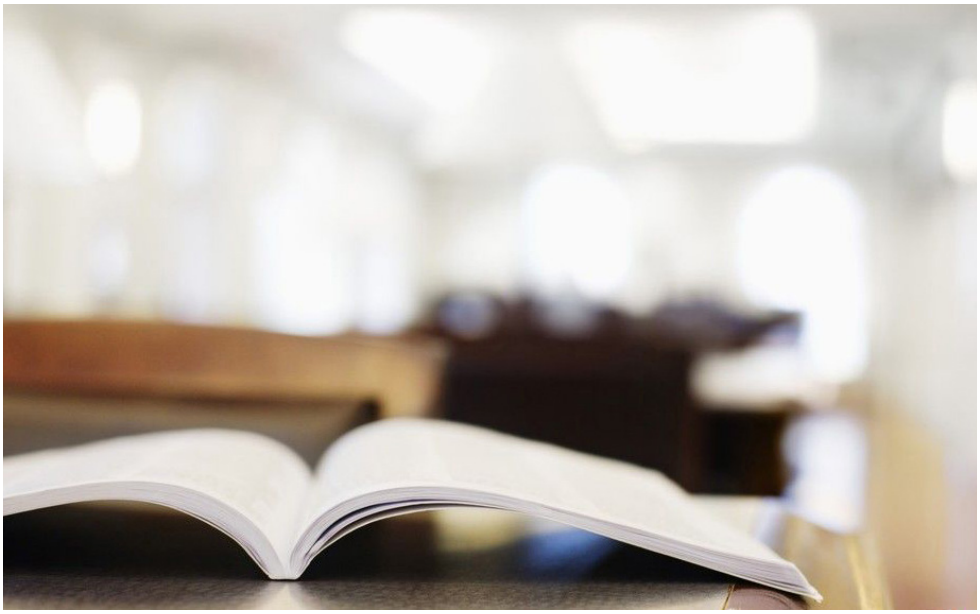
技术人血泪史：七种IT失误让你直接走人

Java 与 .NET 的平台发展之争

Java 8即将正式发布，从早期版本中，我们已经可以领略到一些令人兴奋的特性。但是开发者Andrew C. Oliver表示，尽管如此，Java语言在某些特性上还是落后于.Net。

51CTO.com
技术成就梦想

开发
月刊



出版方：

北京无忧创想信息技术有限公司

责任编辑：

林师授

封面设计：

苍旭

联系方法：

邮箱：linss@51cto.com

电话：010-68476606-8123

出版日期：2013年09月24日

欢迎来稿，请发送邮件至

linss@51cto.com

主编的话

本期《开发月刊》的专题内容是关于面试相关技能，描述了互联网公司的面试经历，给大家分析相关面试技巧，为以后或者现在找工作给出参考。月刊以原创译文为主，把国外最新，最热的技术展现给大家。

专题报道将带你进入那种面试时过五关斩六将的快感，又有人生得失几何的感想，或是让你感觉没有遇到伯乐颇为失望。不管怎么样，做好职业规划，不断学习，提高自己的编程能力和抽象思维能力，相信哪里都有属于你的伯乐。

同时，开发月刊集合了热门的资源下载，最近热门技术专题以及相关开发频道每周重点推荐的内容。不管朋友们喜不喜欢我们推荐的文章，但是我们会一如既往的把开发最好的内容推荐给网友，与网友共同进步，成长。

祝好！

51CTO开发频道寄语

目录 Contents

2013年09月
总第30期

编程排行 Billboard

专题报道 Special report

原创译文 Original translation

技术热点 Techlogy hot

■ 编者按

TIOBE社区今天发布了2013年9月的编程语言排行榜，Transact-SQL 在本期榜单中取得了历史性突破，挤掉Perl、Ruby编程语言首次闯入排行榜前十。前五名内有了小范围的浮动，上个月排名第五的C#这个月排名第六，他的位置被PHP所取代，也算是意料之中的事，因为PHP最近的势头正劲。

9月编程语言排行榜：Transact-SQL首次闯入前十

TIOBE社区今天发布了2013年9月的编程语言排行榜，Transact-SQL 在本期榜单中取得了历史性突破，挤掉Perl、Ruby编程语言首次闯入排行榜前十。前五名内有了小范围的浮动，上个月排名第五的C#这个月排名第六，他的位置被PHP所取代，也算是意料之中的事，因为PHP最近的势头正劲。C、Java保持了排行榜一、二名的位置岿然不动，上个月分别位列三、四名的C++和Objective-C位置互换，不过苹果这个月发布了iphone 5s和iphone 5c两款手机，因此下个月的Objective-C或许能重回第三或更高的位置。

Transact-SQL在本月上升势头迅猛，意料之外确实情理之中。作为Microsoft公司对SQL扩展而生的语言。从一开始T-SQL就拥有了先天的市场优势。

关于Transact-SQL

T- SQL是Microsoft公司在关系型数据库管理系统SQL Server中的SQL-3标准的实现,是微软对SQL的扩展,具有SQL的主要特点,同时增加了变量,运算符,函数,流程控制和注释等语言元素,使得其 功能更加强大.T-SQL对SQL Server 十分重要,SQL Server中使用图形界面能够完成的所有功能,都可以利用T-SQL来实现.使用T-SQL操作时,与SQL Server通信的所有应用程序都通过向服务器发送T-SQL语句来进行,而与应用程序的界面无关。

根据其完成的具体功能,可以将T-SQL语句分为四大类,分别为数据定义语句,数据操作语句,数据控制语句和一些附加的语言元素。

前20名榜单排行榜

Position Sep 2013	Position Sep 2012	Delta in Position	Programming Language	Ratings Sep 2013	Delta Sep 2012	Status
1	1	=	C	16.975%	-2.32%	A
2	2	=	Java	16.154%	-0.11%	A
3	4	↑	C++	8.664%	-0.48%	A
4	3	↓	Objective-C	8.561%	-1.21%	A
5	6	↑	PHP	6.430%	+0.82%	A
6	5	↓	C#	5.564%	-1.03%	A
7	7	=	(Visual) Basic	4.837%	-0.69%	A
8	8	=	Python	3.169%	-0.69%	A
9	11	↑↑	JavaScript	2.015%	+0.69%	A
10	14	↑↑↑↑	Transact-SQL	1.997%	+1.12%	A
11	15	↑↑↑↑	Visual Basic .NET	1.844%	+1.00%	A
12	9	↓↓↓	Perl	1.692%	-0.57%	A
13	10	↓↓↓	Ruby	1.382%	-0.34%	A
14	12	↓↓	Delphi/Object Pascal	0.897%	-0.10%	A-
15	16	↑	Pascal	0.888%	+0.06%	A
16	13	↓↓↓	Lisp	0.770%	-0.20%	A
17	19	↑↑	PL/SQL	0.676%	+0.07%	A-
18	24	↑↑↑↑↑↑	R	0.646%	+0.21%	B
19	20	↑	MATLAB	0.639%	+0.08%	B
20	25	↑↑↑↑↑	COBOL	0.628%	+0.20%	B

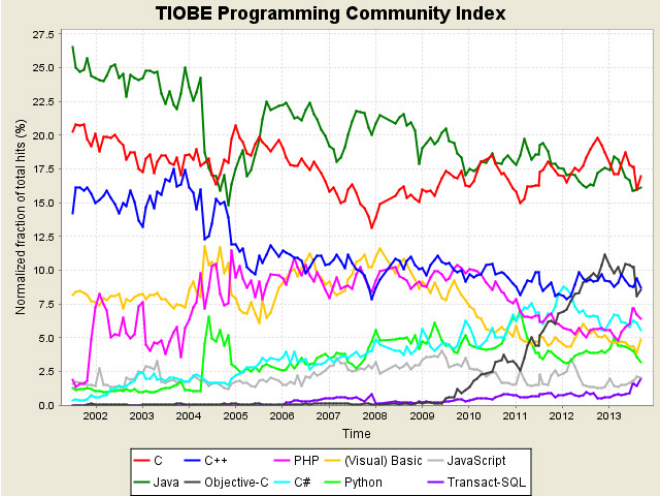
关于TIOBE

TIOBE编程语言社区排行榜是编程语言流行趋势的一个指标，每月更新，这份排行榜排名基于互联网上有经验的程序员、课程和第三方厂商的数量。排名使用著名的搜索引擎（诸如Google、MSN、Yahoo!、Wikipedia、YouTube以及Baidu等）进行计算。请注意这个排行榜只是反 映某个编程语言的热门程

度，并不能说明一门编程语言好不好，或者一门语言所编写的代码数量多少。

这个排行榜可以用来考查你的编程技能是否与时俱进，也可以在开发新系统时作为一个语言选择依据。

前十名编程语言长走势图



后50名的语言：

(Visual) FoxPro, 4th Dimension/4D, ABC, Algol, APL, ATLAS, Automator, Awk, BlitzMax, CFML, cg, CL (OS/400), Clean, Clojure, cT, Dart, Eiffel, Forth, GNU Octave, Icon, Inform, Informix-4GL, INTERCAL, Io, J, JavaFX Script, LabVIEW, Max/MSP, Modula-2, Modula-3, Moto, MS-DOS batch, NATURAL, Oberon, Object Rexx, OCaml, OpenCL, PILOT, Pure Data, Q, Revolution, RPG (OS/400), S, S-PLUS, Smalltalk, Standard ML, VBScript, VHDL, X10, Z shell

21–50编程语言排名：

Position	Programming Language	Ratings
21	SAS	0.627%
22	Groovy	0.612%
23	Ada	0.538%
24	Fortran	0.536%
25	Lua	0.518%
26	F#	0.504%
27	Assembly	0.480%
28	Bash	0.476%
29	Logo	0.434%
30	C shell	0.418%
31	Common Lisp	0.414%
32	ABAP	0.405%
33	Ladder Logic	0.403%
34	ActionScript	0.384%
35	NXT-G	0.383%
36	Erlang	0.381%
37	Scheme	0.374%
38	Prolog	0.373%
39	D	0.331%
40	Tcl	0.329%
41	PostScript	0.311%
42	Scala	0.300%
43	Scratch	0.300%
44	JScript.NET	0.281%
45	PL/I	0.256%
46	Haskell	0.251%
47	Emacs Lisp	0.242%
48	ML	0.241%
49	Go	0.217%
50	OpenEdge ABL	0.206%

编者按

整场小米的面试两个部门加起来共计约7个小时，这是我经历过的最长时间的面试了……小米的面试很辛苦，今天码字也很辛苦，现在已经是凌晨1点半了。

面霸的八月：小米面试记

书接上回，今天叙述小米的面试经历。这里可能有一些技术理解和技术方案，欢迎讨论。另昨天共计收入7笔共95元，够我喝几杯咖啡了，谢谢所有捐钱的朋友。

如果你心疼我码字辛苦，有钱朋友钱场，没钱的请拉朋友来捧个钱场，捧场链接：https://me.alipay.com/chunshengster ，多少不限

小米：运维部

在小米是聊了两个部门的，首先是运维部门，在@wilbur井源 的热情招待下，吃了顿大餐，抱歉的是我没有带足现金，所以付款时我无法“客气”，改天补请。

wilbur井源同两位同事与我四人边吃边聊，我简单介绍当前的网站的服务结构以及部分业务的技术设计，比如网站架构的分布情况，分布式文件系统fastDFS的使用状况、Redis和MySQL的一些部署结构和技术，其中尤其对监控这件事情我做了详细一些的说明（详见服务可用性监控的一些思考以及实践）， 中间提到了关于主动监控（主动监控是指通过运维和业务部门指定监控的系统资源、接口、页面、日志等，主动发现问题,警报级别较高）、被控监控的概念（指通 过JSlib或客户端lib对于所有的操作尤其是网络接口的请求进行监控，对异常进行汇报，通过收集日志的方式进行可用性问题的发现）。当然，还有必不可 少的是对haproxy的运行和优化状况（参见Haproxy配置），MySQL的架构及优化方式（见MySQL架构及运维），Redis常见的性能问题（参见redis架构及运维问题），fastDFS同其他分布式存储MogileFS、TFS、lusterfs的在功能、运维成本上的横向比较，多IDC图片cache的部署以及性能优化（参见多idc图片Cache部署），Linux内核参数（参见Linux内核配置）和让我特别自豪的是关于网卡smp affinity/RPF/RFS的优化效果（参考3/4/5）的一些优化等。当然，这是正经的运维部门，我阐述了我对“运维”工作的理解：60%的分析整理工作加上40%的技能，分析整理能力是做好运维的基础。

井源也询问了几个安全问题，我粗浅的理解是：从系统管理员（SA）的经历来讲，做好IT系统规划，合理区分服务器角色，通过iptables是能够阻止大多数接入层非法请求的；对于web业务的安全来讲，SQL注入、CRSF等攻击是因为对输入输入内容的过滤不严格导致的，在开发的过程中合理使用一些优秀框架或lib，也能够避免大多数漏洞的产生；有个比较有意思的话题是关于溢出的，现在我已经不会计算溢出地址了，在我当script boy的时候研究过一点，忘光光了，惭愧……

井源这边的效率很好，边吃边聊的气氛很放松，不过很多问题都停留在一些思路和效果数据上，没有勾勾画画的太多深入的探讨。

电商部

大约8点半左右到的电商部门，常规面试的第一轮都是技术，包括细节。面试官是位张姓的team leader。

在这轮面试的过程中，因为是在会议室，有笔有板，所以我边讲边写。大体上介绍了我对web服务架构的理解，我认为，web服务架构大体上离不开这样几个层面：接入层（负载均衡）、业务服务层、数据层，一般还会有不少的后台辅助程序进行同步、异步的处理各种不适合在业务层融合的服务单元。数据层可以包括DB、Cache、File等，数据层还可能会有很多中间件或代理服务器用来做数据层的负载均衡或是HA，以及Sharding等。同面试 官详细介绍了当前服务的公司在每一层所采用的技术，分别是：haproxy、nginx+php、twemproxy+redis、 MySQL+RedisCache、Varnish+Squid+nginx+fastDFS。

haproxy的服务器配置是按照100w并发的目标进行配置和优化的，计划100w客户端连接，考虑每个客户端连接可能产生1个内部连接，按照每个连接消耗4k（此处修正为17K，haproxy的官方数据，见参考8,感谢 @GNUer 的修正）内存来算，大约8G(此处修正为32G）内存【这里的计算还需要再考虑，我担心haproxy的每个连接消耗17k内存是包含对内部服务器的连接】，实际上往往比这个数字要大。目前达到的最大连接数目测到过16w，在接入层的系统优化上分别有：网卡中断优化（参考3/4/5），linux 内核参数优化（见linux sysctl.conf配置）。

值得一提的是，我们的haproxy服务器都是64G内存，实际上远远永不到这么多，图片服务的最外层cache，即Varnish，我们也是部署在haproxy服务器。

在最外层服务器上，我们每天大约5亿+（1-1.5亿+的动态请求、3-4亿+的图片请求）的请求量，共计使用7台64G的Dell R410,目前看负载还很低，从系统各种资源上看，请求量翻倍应该是没问题的。

在最外层的服务器配置上，有一个问题值得注意，即sysctl.conf的配置中，timestamp必须为0，这个在tcp协议的扩展标准中有提 到，否有nat环境的客户端连接有可能产生异常，异常的状况可以在netstat -s的输出中看到。还需要注意的是timestamp=0的情况下，tw_reuse是不生效的。

要保证服务器能够接收大并发的连接请求是件不难的事情，但需要考虑一个细节，每接收一个请求，haproxy就需要至少分配一个系统的tcp端口请 求后面的业务服务器、cache服务器，系统一个ip地址可用的端口数最多为65535，一般还需要减去1024。值得考虑的是减 小 tw_bucket 的容量，让系统在tw_bucket满的状况下，对tw状态的连接进行丢弃，以达到快速回收的目的，tw的默认回收时间的2倍的MSL。还有一个方式就是多配置几个ip。

还有一个问题，接入层的服务器往往会开启iptables，内核中nf的相关配置也是需要优化的，比如nf_conntrack_max、nf_conntrack_tcp_timeout_established等。

在业务层的优化有nginx+php（fastcgi连接方式、php-fpm.conf配置中的优化）， 我的一个经验是，如果nginx同phpcgi运行在同一台服务器，采用unix socket的方式进行fastcgi协议的交互是效果最快的，比127.0.0.1的回环地址要快太多。我在08年优化过一台服务器（Dell 2960，16G内存），通过两个步骤，将一台服务器从900qps，优化到6000qps以上,其一是将fastcgi协议运行在unix socket上，其二是合理配置spawn-fcgi的进程数量。现在基本上phpcgi都是运行在php-fpm中的了，其进程池逻辑是我最赞赏的功能之一。

如果nginx和php-fpm不在同一台服务器上，可以考虑使用fastcgi_keepalive的配置，实现nginx同fastcgi服务器持久连接，以提高效率。

nginx+php-fpm提供的运行状态非常有意义，nginx的status模块和php-fpm的status输出可以告诉我们nginx进 程的请求处理状况，php-fpm的status输出可以告诉我们php-fpm的进程池设置是否合理。我们目前对这两个数据通过nagios定期采集，并绘制成图表，很有“观赏价值”。

php-fpm.conf的配置中还有几个参数对优化比较重要，其一是进程自动重启的条件pm.max_requests，其二是php-slow log的配置，slow log 是优化php代码的非常重要的信息。在我目前的环境中，php的慢执行日志是通过rsyslog进行传输并集中分析的，以此反向推进开发对php 代码的优化。

php的服务器在高并发的情况下，有可能因为服务器本身可提供的端口数量的限制，无法同redis服务器建立大量的连接，这时候可以在 sysctl.conf中配合timestamps=1 加上tw_reuse/tw_recycle的方式，进行端口快速回收，以便更好的向数据层建立 连接，接入层的haproxy是不可以这样的。

这一层还涉及到一个安全问题，就是php代码被修改并挂马的状况，我的解决方案是，将php-fpm的运行用户同php代码的属主设置成不同的用户，并且保证php-fpm的运行用户不能对php代码具有写的权限。

数据层的情况里，MySQL主从结构以及MHA+keepalived的高可用配置，这个基本上是看文档应该就能够理解的。如果是5.6的新版 MySQL，其高可用监控可能可以做的更简单，MySQL官方提供对应的工具，只是我还没有测试。对MHA的监控功能，我觉得亮点是MHA对切换过程中 MySQL binlog的获取和执行，在最大程度上避免了数据丢失。但是其缺点也是有的，比如：监控进程在触发切换后就停止了，一旦触发，必须重新启动进程再继续监 控。06年时我在sina做过一个叫Trust DMM的项目，通过 DNS、MON加上自己写的插件，监控MySQL主从集群的可用性，可以实现，主库、主备自动切换（缺乏binlog处理的环节）； 从库是一组服务器，如果从库发生问题，可以自动下线。只是这套系统部署起来比较麻烦。这个项目曾经获得过sina的创新一等奖。

我还提到了我认为的DBA日常的工作至少应该包括：审查并执行上线SQL；定期检查MySQL慢日志并分析，将分析结果反馈到开发部门进行调整；定期审查数据库中索引的效率以及可用性，进行优化我反馈。现在做一个一般水平的DBA已经相当容易了，对percona的工具了解透彻，已经能够解决非常多 的数据库问题了。

MySQL还有一个难缠的问题，numa架构下，大内存服务器内存使用效率的问题，numactl对策略进行调整，如果使用percona的MySQL版本，可以通过memlock配置对MySQL的Innodb引擎进行限制，禁止其使用swap。

MySQL常见的架构里，还有一种主从存储引擎不一致的方式，即主库采用Innodb引擎，提高并发写入的能力，从库采用Myisam引擎，这种方 式目前我们也在采用。这样做一是为了获取更好的读性能，另外是，Myisam引擎的是可以节省内存的。Myisam在索引数据内存读取，数据内容磁盘读取 的状态下，已经可以比较高效的运行了，myisam_use_mmap的配置项，会让MySQL将myisam的data文件也mmap到内存中，这样做 既高效，又可以使用mysiam引擎的特性。

数据库主库要避免一件事情发生，就是无条件删

除和无条件修改，如“delete from table”以及”update table set xxx=yyyy”等无where条件语句，原则来讲是应该禁止执行的，这样的权限不应该开放给开发的同学，甚至DBA都不能无限制的操作。目前我的解决 方案是 sql_safe_updates=1,但这配置是不能够写my.cnf中的，只能启动mysql后进入console进行配置。

当前我们还使用了Redis作为DB，基于主从架构，跨IDC。目前的问题是，复制连接断开后，Redis快照重传的问题，从库会在快照替换期间有 短暂的性能抖动。 Redis2.8新版本psync的特性应该可以改善这个问题。我们还使用twemproxy，目前部署在每一台php服务器上，并监听unix socket，php使用phpredis的模块进行连接。有效减少三次握手的时间。temwproxy还有很多其他的优秀特性，通过一致性hash做 cache集群，可以有效的避免cache迁移问题。通过其对后端redis的健康监控，可以自动下线有故障的redis。

还有针对多IDC的图片存储和Cache部署情况。目前我们自建的图片CDN承载网站每天约4亿的请求，带宽最高峰值约1.5G左右，其结构大体上 是中心IDC存储图片原图+SQUID disk cache存储图片缩略图，在外地IDC使用两级缓存，分别为一层SQUID disk cache（两台，做HA），另一层为Varnish cache（最多四台），实际上，如果仅考虑work around的状态，squid cache层基本上也可以不要的。但是，目前这样的结构可以减少varnish回中心节点的请求，减少中心机房带宽的压力。这个结构还算简 单，varnish在高并发请求下，有一些资源配置是需要注意的，比如NFILES / VARNISH_MAX_THREADS / nuke_limit 等。

沟通的技术问题还是非常多的，包括在井源那里提到监控框架的事情，也尤其提到了我对rsyslog的优化，优化后的rsyslog在可靠性方面是非常值得称赞的（优化思路见参考6）

我有一些将电商三面的运维运维同学的问题综合到这里了，有些话重复的就不再描述。

值得一提的是二面是另一位开发负责人，一看就是个很有独立思考能力的同学，他问了我一个很有意思的问题，大体的意思是，在系统架构方面，有这样的几 个层次，从下往上：使用开源、精通开源，优化并修改开源软件，创造开源软件。问我自己评价我是在哪一个层次的。我认真的思考了一下，我应该是在第二个层 次，有些精通的，有些修改过的。

电商四面是时间最长的，至少有两个小时以上，结束的时候已经是夜里一点四十了，我觉得电商的老大是应该在支付宝里面给我捐一些钱才好的 ，不知道有没有小米的同学能够转告哈 。我们应该是谈到了非常多的事情，包括秒杀的解决方案，包括对持续集成和自动化测试的理解、对后台数据业务类型的开发中数据计算错误的理解，时不时能够得到“我们想的很一致”这样的评价。

那时已近半夜，记忆进入低效态，一些太琐碎的事情记不得了，重复的技术方案也不再赘述。下面简单描述一下我对秒杀的解决方案的理解：10w的数据，从0到10w，不能多卖。目前的问题是，每次到秒杀时分可能同时进入100w的请求/连接。如何破?

我的方案是：排除user、session等外部依赖服务的前提下，两台ha外面抗并发连接（后来想这个无所谓的，不如做成php的服务器），三台PHP服务器（不要使用任何框架，最朴素的纯粹PHP代码），两台Redis（最初说了一台）。具体优化状况如下：

◆ haproxy优化能够支持百万并发连接，这个很容易了

◆ nginx优化worker connections，优化nginx的并发支持能力和请求队列的接收能力

◆ php-fpm优化listen.backlog，优化fastcgi请求队列的接收能力

◆ Redis 假如在秒杀的1分钟内，服务器不出现故障，优化redis的最大连接数

◆ 优化所有服务器的网卡、sysctl参数

PHP的逻辑可以简单理解为对redis的某一个key进行incr原子操作，如果返回的当前数值小于等于10w(两台redis的情况下应小于等于5w)，则为中签。

从我以前看到的数据来讲，redis的最好状态在8w qps。nginx+php在08年时已经优化到6000 qps，目前的服务器设备（双核16cpu+64G内存）达到2、3wQps应该也是不难的事情（这个的最新数据我不知道）。上述配置至少应该能够在5s 内完成10w次redis的incr操作。加上系统各系统对请求队列的支持，可以几乎做到不报错，短暂延迟。

如果考虑1台redis请求量会很高，可以考虑分片，每台分5w。

当然，这是在仅仅思考不到1分钟内给出的方案，从现在来看，haproxy是可以不要，nginx扛并发连接的能力也不错。所有的细节还需要通过压 力测试进行验证。而实际情况加上对其他服务的依赖（我不知到还有哪些，抽丝剥茧去除干扰），方案也会更加复杂一些。据电商老大讲，实际情况是，秒杀的服务 用了十几台服务器，秒杀的时候偶尔出现一些故障，小米做秒杀的同学，压力很大哦。

如果你提到要记录中签的用户的uid和中签号码，还是redis吧。

（突然wps的linux版崩溃了，只能恢复到这里，后面的部分内容是重写的，可能有点混乱）

针对刚才的问题，我在白板上画了个简单的架构图：haproxy+nginx/php+redis，haproxy和nginx/php都是可线性 扩展的，redis可以通过sharding来实现扩展。理论上讲，一个可扩展的架构是可以满足任何性能要求的，更何况如此简单的逻辑，单机性能已经可以 做到非常高了。

电商王姓负责人在问我方案时问这个需求会有哪些难点？我看着白板笑笑：目前看，应该不存在难点。如果有问题，应该看日志和服务状态以及服务器状态。

第四面聊得很头机，对方几次想结束时都突然冒出来一个问题，每一个都会讨论比较久，比如后台的一些计算操作是否换成java更合适，因为java可以更严谨。我说这可能不是语言的问题，而是程序员习惯和素质的问题，如果想换，其实我倒是更愿意尝鲜，如用Go，还可能可以同时满足性能的问题。

还有突然聊到持续集成，我坦言，我对持续集成的理解停留在用工具实现自动测试和发布这样的层面上，没有实操经验。但我个人的一个粗浅的认知是：持续 集成的前提是自动化测试，自动化测试的两个难点：1,自动化测试用例的设计；2,程序员对自动化测试的理解和心理反抗程度。我在目前单位有过短暂的尝试：专业的传统测试人员对测试用例进行设计，程序员接收到的需求应该包括正向逻辑的产品需求和测试用例的需求。开发工作完成的标记是：自己写的测试用例在自己的代码上完全通过，代表自己一项开发工作的完成。

说到这里，对方不禁双手伸出拇指！哈哈哈哈

或多或少也还有一些别的话题，我自认为那晚像演讲一样很精彩，只不过时间已过午夜，其他的一些细节不太记得了，如果想起或小米参加面试的同学有提起，我再补充了。■

面试杂记：三个月的面试回忆录（携程、腾讯等）

■ 话说2年的创业过去了，随着项目的盈利能力越来越弱，基于家庭的压力，不得不暂停创业的步伐，回归到找工作的路程。于是，就有了这两三个月长短性找工作的心得体会。

前话：

话说2年的创业过去了，随着项目的盈利能力越来越弱，基于家庭的压力，不得不暂停创业的步伐，回归到找工作的路程。

于是，就有了这两三个月长短性找工作的心得体会。

前前后后，三个月时间，面试了10家左右，这里就心情的回忆，花个八小时，一次性把所有公司写完了。

以下为各家面试的历程：

朋友推荐：

一大学同学推荐我到一个他旧同事的公司，职位是技术总监了，约了时间过去对方公司聊了聊，地址在体育西那边，感觉聊的不错。

而后归家后，在QQ谈薪水时似乎期望值有点差距，后来没有继续谈下去。

后来过了3星期左右，对方公司觉得真想组团队，于是4次到对方公司里聊，最后一次基本商定在试用15K，转正18K；

可惜天不如人愿，就在第二天去当面确定的时候并商量其它细节的时候，故事有了转折，投资人决定项目二次外包，项目团队延后再组。

同时负责人用了另一个项目说先合作开发，然后说会说服投资人月底组队，结果就给拖到了8月份，还是给放了飞机。

园汇和力方：

由于朋友推荐的那家，基本上被拖着，所以我很少投简历，加上可能由于我简历里写着薪水低于15K勿扰，接下来的2周里，只有两家面试：

一个叫园汇的，地址在天河区马场路那边，是个技术经理职位：去到的时候房子里只有几张桌子和三个人，两个男的和一美女，大体状态是准备做园林项目的电子商务，某园林想走电子商务所以才决定开子公司去做这事，面试我的个技术总监，基本上没啥技术，喜欢做管理，基本上聊的话题，就是问我需要多少人，开发周期多久等，然后缓存，orm也扯了点，主要是想确定我不好管理，专偏技术，这样他的总监才能当的牢，我也表现了更偏技术的一面，聊的还挺愉快，最后问我薪水期望，我在纸上写了15K，然后抬头看了他一眼，他疑惑了一下，我就知道没啥戏了，他只能说考虑，一周内有消息会通知，说明了看我简历的人没看到我那行低于15K勿扰的信息。

另一个叫力方，地址在凤凰软件园的，是个技术总监职位：公司是做GIS相关行业的，面试我的是个开发人员+HR，前后聊了一个多小时，基本上技术和HR都觉的OK，薪资15K也没问题，然后HR对我说，最后只要和副总再聊一下就可以了，结果

人家副总在开会，等了半小时，还在开会，HR无奈让我先回，刚好周五，只好说下周再约见副总，结果，下约无期。

几个小事：

一个是好像是叫桌X公司打来的电话，是个开发经理职位，问了下我期望值15K，职位招聘写10-14.9K，然后说考虑后再约面试。

另一个是好像叫乐天的项目经理职位，虽然发了邮件邀请面试，不过连个电话也没打，加上面试的时间是周五或周六，估计是六天班制，就没去。

有几个是搜到我简历后主动打电话过来的，不过职位都是开发职位，就被我谢绝了。还有一个是游爱的，公司是搞游戏开发的，好像园子里还有它的招聘信息，不知道怎么的也看上我了。

优迈和聚星源：

大概是感觉面试的概率太小，于是经常性的调整简历，把薪资改成面谈，去掉一些约束性文字，自我介绍信息也改成谦虚型的，结果很快就有了两个面试：

优迈，地址在体育西路109号高盛大厦，是个开发经理职位，前台MM是个冷美人，填了资料，还有一份心理测试题，基本是走路是走的慢还是走的快，是抬头望天型，还是低头思乡型，没事的时候是双手交叉还是双手放平，类似这类的。

写了写交卷后，等了一会来了另外一个人，从8楼被带到11楼面试，来了2个技术人员，一个职位是属于开发人员，另一名不清楚，基本上聊的话题，就是把我简历上的项目及项目的实现原理给详细的介绍了一翻；第二个人聊到分布式负载压力测试工具时，聊到了导出分析报表的问题，我说工具只负责产生压力，详情在性能的分析还是在要服务端上分析，对方说没听说过性能要到服务端分析的，还举了LoadRunner负载测试后有报告，从没告诉人家说性能还要跑服务器里分析。我只能呵呵，考虑到在这里反驳可能会不太愉快，毕竟是面试不是同事的技术讨论。

后来聊完，还是要写待遇15K上去，然后就是等一下，他去找人事，结果回来就说人事不在，说下次再约，结果就是无下文了。

聚星源：地址在科学城 科汇金谷，职位是开发项目经理，去了之后直接就变项目经理， 前台MM很美也很友善的说，一去到就是填资料，还有做C#基础面试题，我是对做题有点意见的，不过人家MM她们公司就这流程，看在MM面子上，我就安静的做题了。我嚓，连WebForm里的验证控件指定正则表达式是哪个属性名称的填空题都有，凭着个人的一线开发经验，基本上算全答了，花了三十分钟左右，然后就是无情的等等等，光等面试官和HR的时间就差多不一个小时。

N久后，终于等来了面试官，是个小胖，听说是个经理，基本不聊技术，聊起了我以前做的一些项目和项目管理的内容，还有这职位可能出差的情况，是否能接受之类的，聊完后，写完薪资，等了N久快下班了，一个略有丑感的HR才跑进来说：快下班了，我们聊快点，你来的太晚了。我嚓，在你们公司都呆了三小时了，等了我都快起毛了，于是表情大伙都很急，接下来HR一来就说，你管理方面的知识有点弱，我急着解释补充知识，说的有点乱，最后HR说觉的你的期望有点高，问最低多少，我嚓，砍价啊~~~ 本来就不爽，遇上这种先说你这弱那弱再砍价的HR，基本上谈不到一块，很快我就急着闪人了。

广凌和众通：

接下来两个，都是软件部经理职位：

广凌：地址在天河区五山路248号金山大厦，打电话来的时候，由于是手机号打来了，而且声音像极了我某位网友，我以为开玩笑的，疑或了一下，多问了几次对方公司的名字，最后对方说没兴趣就不要浪费时间了，后来我还是约了第二天面试，并让她发封邮件到我邮箱，可能对方不爽，说直接发短信好了，后来连短信都没等到。

不过第二天我还是去了，一去就填资料，还要特详细的那种，查户口都没要求写那么多，然后是一张画图题，不知道要考察啥，把一个形状向四方再延长画，HR是个美少妇，可惜我留了点胡子，头发也还算长，形象有点苍桑，没给对方最帅的状态，后来聊完，基本上算查完户口了。

对方又拿了一份题过来让我做，又是C#高级面试题，虽然不爽，不过基本上半小时也全答完了，然后交题。然后说等部门经理过来面试了，已经有经理，还招经理？问了下HR，HR说那个人要离职了，招人来顶位。

过了一会，来了一个中胖，不聊技术，直接就聊管理，项目管理基本上一略而过，主点是聊人的管理，我嚓，第一次聊到这类话题，心虚啊，顿感自己渺小了很多，最后是被完败了。

众通：地址是珠江新城南天广场，是一家小创业公司， 七八个开发者+一个BOSS。在等BOSS面试的时候，一个小青年跑过来问我是不是面试，我说是，然后他问我会不会JS，我瞬时和我的小伙伴惊呆了~~~这，我只好说懂点基础，估计被鄙视了~~~

接下来就和BOSS聊了，由于我创过业，他正好也在创业期，所以对我还是很感兴趣的，基本上是一个平等的面聊过程，他问我的和我问他的，基本上一样多。

除了展示自己的技术能力外，基本的话题都在公司的战略方向上谈，也谈一些创业期会遇到的问题。

之后也报了薪水，对方说觉的有点高了，我说这要看从哪个角度或方向上看了，如果是XXX的方向的，价值会是报这个数的2倍以上。

之后约了隔一天的周六再深谈，周六去了，他正好在给那问我会不会JS的同学做思想教育。

刚好有个人去面试攻城师，我就和那小青年聊聊天，给他介绍下并演示了我的框架，过了半小时，那人面试完，我就进去和BOSS又深聊了一次，基本上确定了方向，七八个开发者归我管，他走营销。最后就是谈待遇问题，对方说给了8-9K，年底盈利10%分红，3年干满给6%股份。

基础薪资我觉的有点偏低，要求12K，毕竟分红和股份，似饼一下的承诺，结果对方说考虑，下周给答复，结果下周就是无下文了~~~

德捷：

地址是东华西路八十六号丽华大厦，职位是.NET架构师，HR打电话来的时候，基本上先问了英文怎样，平时使用的技术是什么之类的，和我的期望薪水，然后就约了第二天面试。

去到后，先是填写资料，然后是做一篇英文题，给了一本大中华英文词典，只有两大题（6小题），一题是英译中，一题是中译英（5道题）。

英译中，是讲的CLI的相关基础介绍，看着那破旧的大词典，我感觉到它被被多少人曾经压过~~~找单词时，也被人用笔划出来了。

然后是中译英，基本是测试某某用户的某某权限之类的，可惜字典无法中译英，只能靠意淫翻译，前后花了30分钟左右。

英文题后，是上机题，是考察设计模式的，基本就是两个表单，要求用模板模式给折腾出来，然后是一道要求适配器模式实现的题，前后花了四十分钟左右。

然后是一个架构师经理的人员过面试我，先是花了5分钟给对方讲上机题的思路，然后就是房间里聊天了，聊了1个多小时。

对方很友善很直接，先是聊技术，再聊产品，再聊设计，然后也说到职位规划。

对方说这个职位，就是技术里最高的了，公司就三十来个人，基本上也没啥职位可以往上升了，架构师之上就是他了，他之上就是一个副总，副总上就是老板了，除非他辞职， 不然也没啥发展了。

对了，公司是做以合同管理为主的信息类管理系统。

最后都是一样的步骤，报个薪水就可以等通知了，不过这一家，是唯一一家有电话通知对方是否被录用的，虽然来的比承诺的晚了一天。

刚好电话来的时候，我人不在，归来时，发现两个未接电话，然后是一条短信：因未能达到招聘要求，所以未能被录用，感谢关注XXX。

携程：

携程：开发经理职位，园友推荐了携程无线事业部，说以我能力薪资在20-30K+可谈，由于地址在上海，所以先电话面试，以前没来过电话面试，所以电话面试还是有点期待的。

一开始来的是个小伙，电话一来，基本就是面试宝典题，只记得一来就GC，忘了其它题了，后来我引导了下他，之后就聊到我简历的项目上来了，所以总体还是聊的不错，感觉这小伙还是不错的。后来HR说二面要直接过上海，但又不肯报销路费，所以也不打算去了。

后来沟通下还是先尝试电话再面一次，觉的把握高了再看看，于是有了二次电话面试。

二次电话面试，这是个悲剧，本来约好下午一点的，结果对方忘了还是没空，又推到了对方下班时间，一来，又是面试宝典，又说GC。

不知道那两哥们青年是不是整天研究GC的，还是看的是同一份宝典。

后来问了下他知道我面试的是什么职位不，有没有看我简历，对方说他只是负责面试，不知道我面试啥职位，也没我简历，我很悲催，只有任他拿着宝典找题抽我。

刚好有几个记忆题被他抽到了，我脑袋卡了一下，说不知道，你给我说一下，对方说时间有限，不解答。

后来我让他问点别的，没想到他竟然问设计模式，一来就单例，我觉得这类问题意义不大，告诉对方某人的博客里有有单例的五种实现方式，对方说不知道，只要我回答一种实现，悲催的我只好说了构造函数私用，内部定义一个静态实例对象的常规解答，没想到对方又让我讲讲其它设计模式，像什么命令模式之类的，我就嚓了，我也忘了当时怎么答的。

基本上剩下的点时间，我就问了一下他部门的一些状况，大概也问了十五分钟。

后来基本上也有结果了，被坑了，HR反应结果不好，说如果人在上海可以过去实地聊聊，如果人不在就算了。

后来我就反思，你总不能不让人家这么问，为避免再被小弟坑，我认真去看了一遍面试宝典题。

腾讯

腾讯：是个T3级别的C#高级攻城师职位，推荐人跨度比较大，至少25K+，冲着这个待遇，我就从广州去了深圳面试。

第一次面试其实是电话面试，聊没几分钟，对方说电话说不清，让我直接过深圳面试，于是周日就过深圳了。

到了深圳，周一，对方大概没空，约了对方下班时间17：30面试，在松日大厦里，一开始来了两个人，一个架构师和一个开发经理。

架构师：基本上的技术问题都是他问的，总体上问的还不错，先简单扫了我简历，然后从项目入手，再深入到一些具体的项目，总体上有问有答，技术上架构师也没啥可问，都是明白人，然后他就先出去了。（从最后一句我说到hash取值对方停顿疑惑不解的时候，加上他问的问题基本上和.NET无关，我猜测他应该是其它语言调进.NET部门的。）

开发经理：就问我擅长什么，我就答了.NET领域的基本都擅长，问我前端怎样，我说前端一般。（基本就问你会不会前端）

PM：人好，一来就自我介绍，并介绍了前面2个人的身份，有没聊技术我忘了，但记得有问了我一些创业期的内容，和给我解答了小组的情况，最后是要等小组最后一个人面试，结果最后一个人在开会，没空，约了第二天再去一次。

Leader：第二天暴雨，我还是去了，去到后，PM去叫了一个带着Apple笔记本男进来，看着他年轻，我以为是小组内的一个开发者，一来就说：说点你最厉害的技术，我瞬时和我的小伙伴惊呆了，这么泛的问题，我说我技术比较泛，也写过很多文章，一时也想不出哪个最厉害，他说，那你找一篇你写的最好的文章讲讲，我再一次惊呆了，我说我写的文章很多，然后呆了几秒，还是给他讲了一个AOP的应用的，结果他估计完全不懂，于是他说还是打开我博客，看了我博客后，说我文章写的挺好的，就是表达沟通能力太弱。

后来他问：网站运行慢，怎么去定位问题。我给他讲了大体的分析流程，从前端到后端，从逻辑到数据库，各层次截断分析，后来又给对方普及了分布式缓存的相关内容。

最后对方笑着说，你技术很强，学习能力也很强，就是感觉缺了点什么，最后崩出三个字：缺少点感染力，说高级攻城师还是需要具备感染力的，虽然后来我一再解释，不过估计对方思维定式了，也没折。

正常听说腾讯的面试官是不会问工资的，不过这个Leader倒是问了我期望值，我把推荐人的起薪水25K+报给了对方，然后对方说和小组再评估一下情况，让回去等消息。

基本上到这里，就没消息了，后来听推荐的人说他问过那个开发经理：觉的我虽然后端强，但前端一般，不太适合他们的项目技术（内部OA）。

我当天也主动发了短信问PM后续有安排没，收到的回信是等一周，OK再通知，结果大伙懂的。

后续有遇到一个网友，他说也去过腾讯面试，只见着PM一个，面试在基础题就被刷了，不过他

之前6月才去面试过电子商务的，只要12K就被HR给挂了，所以他觉的这职位不可能有25K，我和我的小伙伴又一次惊呆了。

赛酷比：

今天去面试的，职位是系统架构师，人在深圳的时候来的电话，还问了我期望值，我说15K，我反问对方说，你们公司公司的期望值呢？对方人事说，之家有个人面试，只要了10K，不过那个人工作年限没我长，只要有能力，公司这个价格是可以给的。

去到的时候，对方人事不在（等到我面试快完了，才打电话问我有没有过去面试，她人在本部），然后一个女开发者临时充当了前台，找了一个资料填写的和一份系统架构师的题目，看到题目时，我差点跪了，有一种想立即走人的冲动。

一共20道题，全是解答题，前五题是.net3.5和4.0的新功能比较和一些4.0并发编程的概念性问题，然后是3道asp.net MVC的题目，还有事件顺序题，领域驱动题，和场景解答题，太强大了。

认真扫了两遍题目，假如认真分析认真做题，基本上我做完对方都下班回到家了，而且按答卷若评分，怎么答也是不及格的。

而且这样的题目，基本一看除了个别题目和公司的项目有点关，其它就是装B用的。

于是我就写上了“已阅”两字，后来觉的既然来了，还是要认真答题：前五题，我就写了“以上五题详见搜索引擎”，后面的题基本就是精要式的。

比如有一题：缓存，NoSql，消息队列在大系统中的作用“，我就精要的答了两个字”重要“。

还有用领域模型设计模式写一个大数量下复杂订单的处理模式和一些的接口及XX参数之类的，我精要的答了现阶段领域模式不适合复杂的业务。

其实重点还是要理解对方的意图，理解后，再用精要式解答，发现30分钟左右就解决完20题了，然后就交卷了。

后来有2个开发者来面试我，一般来说，下级面上级，我是有点意见的，一来对方没有决定权，却有反对权，二来对方的思维和自已差了几个层次。

不过我也扫过面试宝典，所以在技术上不管是对方是何种层次的人员，基本上能问倒我的可能性为1%以下。

不知道两个开发者看了我的答题，有什么样的反应，估想总会有点意见。

后来就开始聊了，悲催的是对方竟然没有我的简历，只能从我手工填写的简要信息里问。给对方介绍了自定义的mvc模式，然后聊对方的项目的重构问题。

给对方的要进行的项目重构一些意见，后来好不容易是开了电脑，上了51job才把我简历弄到电脑上，对方能看一下，不过这时候对方又叫了一个不明身份的人来参与聊，年龄不小，应该是个高层，基本上话题，都是围绕在在对方的项目如何实施重构的问题。

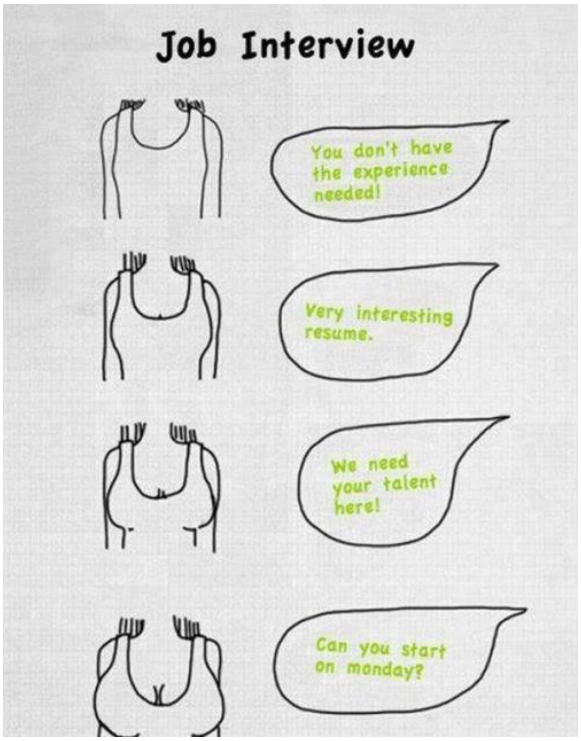
聊着一半，对方那人事来电话问我有没有过去面试，我说我就在你们公司面试着，听完电话回去，对方就结束了，让填写基本薪水，又是下周有消息就通知复试，然后就归家了。

总结：

我也不好传播消极论~~~你的评论，就是我的总结。■

我拒绝参加你们的技术面试

■ 我十分不擅长对付技术面试。就我参加过的几十次来说，我甚至连一次都没有通过。一般来说，过程是这样的：因为一次偶然的联系，一个经理或一个招募者……



我十分不擅长对付技术面试。就我参加过的几十次来说，我甚至连一次都没有通过。一般来说，过程是这样的：因为一次偶然的联系，一个经理或一个招募者突如其来的给我打来电话。之后，我与招聘经理进行一次电话约谈，一般以对方非常满意结束。最后，我当场（现在一般是线上）与技术人员们深入讨论技术细节，之后被淘汰掉。

有时，他们提出的问题我根本不了解。其他时候，我甚至对自己非常熟悉的话题也哑口无言。（有一次我竟然连我最爱的电子游戏的名字都说不上来。）我常常在一些逻辑问题上表现的很差。我真正得到的工作，每一个都是因为有朋友帮忙。

好些年我都生活在害怕面试的阴影中，因为我认为我不会成功。但同时，我知道我自己其实是一个非常优秀的开发者。我总是团队的核心，常常一个人搞定大型项目，并且能够成功胜任领导者的位置。

我对自己面试的问题焦头烂额，为此，我模仿我经历过的那些面试建立一个面试流程。包括脑筋急转弯，测试，技术质询，这些所有过程。当我简略的观察过 包含了大量雇佣记录的流程之后，我了解到，在一个重要的雇佣流程中起决定性作用的是这个参与者是否已经认识雇佣团队里的某一个人。虽然你不能简单的就相信 一个雇员说他的这个大学哥们非常牛，但你也不用担心让他来担起整个雇佣这个人的责任。那么，作为应聘者，你该做些什么?

终于，在一个小型启动项目的面试中，我近乎艰难的用另一种途径找到了解决方法。我首先和项目负责人共进午餐，接着跟团队里的所有人都聊了聊。我们讨 论了一些技术方面的问题，但他们没有尝试或者审核我的技能。取而代之的是他们给了我一份付费合同，让我去做一份他们真正需要做的工作。他们给我恰到好处的 指点好让我开始，接着让我回去在自己的设备上工作。在提供良好的交流的条件下，看我是否能够按时做好这项工作。我晚上花了10个小时就搞定了它。三天之后 我得到了这个工作!

从那之后，我就拒绝参加传统的技术面试了。

我礼貌地建议：用一个短期合同工作去衡量一个资深开发者，这可能是最好的选择。当他们不了解你时，这样十分有效。如果他们真的需要你，那效果更好。此外，另一个好处在于，在跟团队合作之前你就已经见识过他们如何工作的了。

有一些公司拒绝使用我提出的模式，对此我非常理解。这些工作，我不去理会便是。对他们表示感谢之后，我就开始准备下一个工作了。

用这样的方式成功通过面试会给你带来一定程度的信度和影响力，这是传统面试所不能提供的。我用这种方式参加面试的通过率是100%（4次全中）。我最终接受的那个公司，不仅雇佣了我，还打算为我投资一项新的启动项目。

在传统面试上表现的很好的人，你们应该继续参加对你们有利的传统面试。但是，我力荐任何一家公司仔细思考一下，你们的面试流程到底在筛选什么样的 人。真的选出了能够完成高质量的工作并且与团队配合良好的雇员吗？是否只是选出了那些曾经听过你提出的脑筋急转弯的人呢？面试官是否只是在面试时敷衍了 事，之后再去征求其他人的感受呢？或许那个经理非常善于察言观色，但他离开了之后又怎么办呢？好好想想，用短期的合同工的方式能否提供一个对面试者更好的 评价吧。

作者后来的补充：

感谢所有感兴趣的人以及medium、twitter和hackernews的读者们。我没想到居然有这么多回复。我猜大部分工程师都有过一两次糟糕的面试经历，也可能不止。

我今天读到了很多不错的批评，我简要的澄清一下并在此回复他们。

老是不停的问我什么事闭包以及可变关键字代表什么，这不适用于以对话为基础的面试。的确，了解我是否会使用CSS（我不会）或者我是否知道A*算法 （我确实知道）是非常重要的。但我非常请求取消手写代码，某些脑筋急转弯以及现场编程等等。就我的经验，如果我们讨论的时候有一杯啤酒或者威士忌，我会非 常高兴。

我并不是一个没有存在感的程序员。我曾在游戏领域担任过策划者，做了无数的团队发言，迎合了不少投资者，还有许多会议演讲和出版商洽淡等等。这些事 情与面试的不同在于，我基本上总能说：”我需要回去考虑一下。我会明天再联系你。”对方也会说：“我现在可能还不清楚。”我已经掌握了如何准备和预演，但 必须在我对从代码深入到准备好的对话选项这个过程感到自然了之后再开始。因此在技术面试上我就没法做决策了。

最重要的是，你根本不知道什么时候就走进了一个技术面试的圈套。要说有什么不同的话，一个优秀的面试者知道判断一个参与者的好坏正是面试官的工作， 而非自己的。但是，优秀的面试者少之又少。一旦你被一个愚蠢的脑筋急转弯卡住，就像踩进一个捕兽夹一样，你就与这份工作无缘了。跳过踩进雷区这一个环节， 我就能进入我另一个成功几率更大并与团队合作良好的状态。

再次感谢你们的阅读和评论。

我之前的一个同事问我，能在面试时拒绝回答面试官提出的某一个问题吗？我本人在面试时从未拒绝回答过任何技术问题。如果我自己处于那种情况，我觉认为，直接拒绝甚至都不尝试一下是非常不尊重他人的。而作为面试官，如果参与者拒绝回答，我会觉得他甚至连一点思路都没有。

避免被一些技能非常有限的面试官随意摆布，这是我们的目标。■

今日面试题灯：及数组统计分析

有100盏灯，依次编号1-100，初始都是关着的。第1次遍历，打开全部的灯；第2次遍历，关掉第2盏、第4盏等被2整除的灯；第3次打开被3整除的灯；第i次，对被i整除的灯做如下操作

如果灯开着，就关掉

如果灯关着，就打开

如此交替进行，知道100次遍历完毕，请问，还有多少盏灯亮着。

数组统计分析

原题

给定数组A，大小为n，数组元素为1到n的数字，不过有的数字出现了多次，有的数字没有出现。请给出算法和程序，统计哪些数字没有出现，哪些数字出现了多少次。能够在O(n)的时间复杂度，O(1)的空间复杂度要求下完成么？

分析

这个题目，是有一定技巧的。技巧是需要慢慢积累，待经验多了之后，可以灵感或者直觉，就产生了技巧。如果不知道技巧，那该怎么办呢？ 在开始分析之前，说明两个问题：

原数组是没有排序的。如果排序了，很简单。

O(1)的空间含义，可以使用变量，但不能开辟数组或者map等来计数。

这个题目，很直接的解法就是两层遍历，O(n^2)的复杂度，O(1)的空间。空间满足了，但是时间没有。 很多类似的题目，都会用XOR的方法，大家仔细想一下，这个题目，可以么？或者这个题目和可以用XOR的题目的差异在哪儿？最直接的就是，每一个数字的重复的次数是不同的。

还有就是以空间换时间的方法，例如用hash map或者数组来计数。时间满足了，但是空间没有满足。 那怎样才能有时间复杂度O(n)，空间复杂度O(1)的算法呢？不能开辟新的空间，那么只剩下，重复利用数组A。那么该如何利用数组A呢？

首先，我们介绍一种三次遍历数组的方法,我们都考虑数组从0开始：

- 第一次遍历：对于每一个A[i] = A[i] * n
- 第二次遍历：对于每一个i，A[A[i]/n]++
- 第三次遍历：对于每一个i，A[i] % n就是出现次数

A[i]应该出现在A中的A[i]位置，乘以n、再除以n，很容易的来回变换；第二次遍历，对于A[i]本来所在的位置不断增1，但绝对不对超出n的，那每一个i出现的次数，就是A[i]对n取余。

还有一种两次遍历的方法，也是上面的思路：题目中数组是1到n，为了方便算法考虑，以及数组存储方便，我们考虑0-n-1，结果是相同的。 考虑A[i]，现在位置是i，如果采用A来计数，它的位置应该是A[i] % n，找到计数位置，该如何处理这个位置呢？加1么？显然不可以，这里有一个技巧，就是加n，有两个原因

- 加n可以保证A[i] % n是不变的
- A数组，最后每一个元素表示为A[i] = x + k*n，其中x

上面的思路，转换为代码如下：■

```
void repetitions(int A[],int n)
{
    int i=0;
    for(i=0;i<n;i++)
        A[A[i] %n] + = n;
    for(i=0;i<n;i++) {
        frequency = A[i]/n;
      element=i;
    print("Element= %d , frequency = %d", element, frequency );
  }
}
```

技术人血泪史：七种IT失误让你直接走人

□ 核子可乐译

IT人士的真实故事：搞出大麻烦，旋即遭解雇

如今想找一份理想的IT工作并不容易，但丢掉一份工作却非常简单。

导致自己被炒鱿鱼的原因很多，无论是没能尽到保护雇主数字资产的义务、或者是滥用手中的权限以达到自己的邪恶目的，我们都将因此跟自己的职业生涯挥手道别。在错误的时间大放厥词或者在正确的时间闭口不言都会造成严重后果。打探老板的隐私、向雇主说谎或者由于自身的直接原因造成数百万美元的停机损失，这一切疏忽都将把我们的仕途引向深渊。

在某些情况下，每人能找到正确的处理方式。然而某些失误却会引发致命的影响，就算没有因此丢掉工作、大家在余生中也不用指望获得提升了。

在本文中，我将与大家分享七个发生在IT人士身上的真实故事——当然，他们都已经被扫地出门。为了保护他们的隐私，我们将以化名相称。但最重要的是，千万不要让他们的悲剧发生在您自己身上。

致命IT失误一：疏于备份

这是某个周四的夜里十点半，Eric Schlissel的电话响了起来。电话那边响起了某家中型服装制造商的首席运营官的声音，而且在此之前Schlissel从没跟他打过交道。这位运营官在谷歌引擎上找到了Schlissel的电话，而他现在听起来似乎有些狂躁不安。他车间中的ERP系统受到了病毒的破坏，而他需要在第二天早上之前保证一切恢复正常。

作为管理服务商GeekTek IT Services公司的CEO，Schlissel跳上自己的车子全速赶往位于洛杉矶的这家服装工厂，打算亲自处理这一紧急情况。

“经过三分钟的尝试，我发现自己根本没法正常登录，因为服务器上的内容已经荡然无存，”Schlissel表示。“所有数据文件、数据库以及ERP软件都不见了踪影，我只能实言相告——这根本不是病毒造成的。有人把系统彻底清空了。”

事实证明，是某位心怀不满的IT承包商以此为手段对这家服务厂商进行报复，但更糟的消息还有后面——整套备份机制已经很长时间没有正常运作了。Schlissel所能找到的最新数据已经是一年半之前的，这使得备份内容几乎毫无价值可言。

这家服装厂商的最后一线生机来自财务部门。某位负责会计工作的员工一直对IT技术持怀疑态度，因此将所有往来记录都抄写在一份纸质文件当中。Schlissel和他的团队花了六个月时间才会所有数据恢复到服务器当中。

“这是一家总值达1000万到1200万美元的企业，而这次事件给他们造成的损失达到约200万美元，”他解释称。“这也是我职业生涯中所遇到过的最惨烈的IT事故。”

负责这家工厂备份工作的IT人员完全把自己的任务抛到了一边，于是乎他在第二天就遭到辞退。

备份机制未能正常运行的状况极为常见，而这种失误对于个人职业生涯来说无疑是致命的，Schlissel指出。

“我们与新客户接触后的第一件事就是检查其备份情况，”Schlissel指出。“这是IT领域的一类经典事故，我们常常提醒客户，这绝不是危言耸听、而只是确保其固有资产得到严格保护的必要步骤。”

故事寓意：备份在手，万事无忧。

致命IT失误二：探听老板的隐私

就在几个月之前，南加州某家中型医疗保健供应商的CFO给Oli Thordarson打来了电话。作为高级网络管理服务企业Alvaka公司的CEO，Thordarson和他的团队经常需要以临时CIO的身份为小型企业提供服务、负责处理证据调查工作。

这位CFO告知Thordarson，他怀疑有人在偷偷查阅他的电子邮件。事实上，他已经猜测了嫌疑最大的对象：IT主管。

这位CFO表示，在过去两年中，IT主管多次针对他根本没有接触到的业务发表意见。Thordarson告诉我们，“这位IT主管似乎比企业内的任何一位成员都更了解当前的业务走向，这实在是个极大的讽刺。”

Thordarson和他的一位技术人员修改了一项实时网络探测指示器，这样一旦有人阅读了那些自己本不该看到的邮件，指示器就会悄悄发送警示信息。几天之内，Alvaka发现IT主管确实在查阅CFO的邮件——甚至连CEO、董事长以及其他高层的往来信息也未能幸免。而就在第二天，IT主管开始查阅Monster.com网站上的招聘资料。

Thordarson补充称，这类问题的出现频率远高于大家的想象。在Alvaka接触过的企业中，有约三分之二发生过技术人员阅读雇主邮件的状况，其中包括高层管理者的信息。

“他们这么做是只是为了更好地提供支持服务并忘记撤销自己的操作，还是在刻意窥探老板的隐私？”Thordarson指出疑问。“我们也弄不清楚。”

故事寓意：这是个傻瓜如何快速丢掉工作的故事，别无其它。

致命IT失误三：掩盖自身罪行

每个人都会或多或少犯下一些错误。一家大型金融机构的IT人员打算更换一套陈旧的存储阵列托盘。一位工作人员与供应商取得了电话联系，并确认货品已经发出。然而供应商处的初级销售人员发错了托盘型号——该托盘只适用于新机型，无法与旧机型相兼容。

由此引发的阵列故障带来了灾难性的后果，整家银行的业务系统在近一周时间内始终处于脱机状态，造成高达数百万美元的交易损失。手足无措的管理者只能打电话给Anthony R. Howard前来进行故障排查。

根据Howard（他是畅销书<看不见的敌人：黑狐>的作者，同时也担任财富五十强企业及美国军方

的独立技术顾问）的分析，这家金融机构共存在三大问题。当然，供应商发错设备也是原因之一；但另一大失误在于，银行的IT人员没有静候厂商的专业技术人员前来安装、反而决定亲自动手。

第三个也是最严重的问题在于，几乎每个与这次事故相当的人员都选择了谎报军情，Howard告诉我们。只有一位职员勇于坦言当时的真实情况。

“当IT人员发现自己的工作可能命悬一线，肯定会想办法保护自己，并把责任推到供应商方面的技术支持人员头上，”Howard指出。“银行内部团队在完成调查后发现，只有一个人勇于承认现实，而他也是惟一一位在事故后保住了工作的员工。”

故事寓意：也许失误本身并不会让你失去工作，但一味掩饰会让情况变得更糟糕。

致命IT失误四：色情内容

几年前的一个深夜，某位效力于财富百强企业的网络管理员想找出一份空白的备份磁带。他从某位高级系统管理员的抽屉里随便拿出一盘并塞进驱动器——但令人意外的是，磁盘里已经塞满了数据。到底是些什么内容？好奇的他打算一探究竟。

相信大家已经猜到了问题的答案。

“里面塞满了色情内容，”Foreground Security公司总裁兼CIO Dave Amsler告诉我们，当时正是这家公司介入到事件的调查当中。“我们当时检查了这位管理员抽屉中的其它几十盘磁带，但没有发现其它违法内容——谢天谢地。不过就算是这样，该管理员也被当场辞退。”

但在Foreground为各大美国企业及政府机关提供安全管理服务的十四年中，这还不是Amsler亲身经历的最糟糕的情况。Amsler表示，至少有十家客户打电话请他去处理与色情内容有关的事件。有两次，他甚至发现IT管理者在企业服务器上运营成人网站。在这些案例中，管理员们似乎乐于在工作之余利用企业资源为自己安排一点特殊爱好。

色情过滤器之类的机制根本起不到任何作用，因为IT人士们很清楚如何把它们关掉。即使企业拥有严格的管理政策以及落实到位的过滤机制，高层管理人员仍然能利用自身权限从中找到可乘之机，Amsler表示。

“有时候这种情况也属必然，”他补充称。“一般来说，高级管理员有资格访问那些长期处于封闭状态下的站点以完成工作，但这并不代表他们可以彻底摆脱监控机制的掌握。即使是最善良的员工也可能在缺乏监管的情况下做出令人意外的举动。如果管理员意识到自己正处于监控之下，其行为必然会得到很好的约束与规范。”

故事寓意：有些事情最好还是在自己家里做。

致命IT失误五：保守错误的秘密

直到最近，Dana B.一直在某家主要美国互联网供应商担任网络工程师。有一天，一位前任同事被要求变更一部分生产路由器的IP地址。由于这些变更可能影响互联网订阅服务，因此需要将这些服务暂时切换为离线状态——互联网服务供应商通常会在深夜执行这类变更。

但这位工程师不想把大好夜晚浪费在公司里，因此他在下班回家之前就提前改变了IP地址，然后关掉了手机——这样就没人能破坏他美好的业余时间了。

这只是他犯下的第一个错误。更糟糕的是，他一直拒绝对自己的操作内容进行记录，Dana表示。这意味着其他员工根本不知道他到底使用了哪些IP地址。

在他离开之后，工作人员发现接口无法正常连通，因为其原有IP地址已经被占用，这直接导致近5000项订阅服务无法访问互联网。而其他工程师给他打电话希望弄清楚问题到底出在哪里时，却发现这家伙关掉了手机。

“我们组织了一个由五位工程师组成的网络技术团队，花了好几个小时才发现并成功解决了问题，”Dana回忆道。“第二天，他像平常一样走进办公室、又很快被赶了出去。”

故事寓意：有些秘密最好早点公诸于众。

致命IT失误六：严酷无比的灾难

有些人以为自己已经针对可能出现的状况做好了充分准备。某家从事高度监管行业的企业曾花费数百万美元建立起一套全面的灾难恢复计划，甚至包括一家容纳着数百套虚拟机及千兆以太网连接的专用故障转移数据中心。

但一次意外的网络中断彻底切断了主数据中心与故障转移中心之间的连接通路，在整套灾难恢复解决方案身上投下的巨资也瞬间打了水漂。

“CTO并没有确保灾难恢复计划奏效的把握，因为技术人员从来没有进行过测试，”SunGard Availability Services公司恢复服务产品管理副总裁Michael de la Torre解释称，这家公司在本次事件中负责为当事方提供灾难恢复策略。“相反，他坐等了一天多，希望线路能被尽快修复。在此期间，所有用户都面临脱机状态，员工无法通过电子邮件进行交流或者访问数据文件，这直接导致企业的信誉受到严重冲击。”

此后不久，这位CTO的职业生涯也随着此次计划外停机而一同陨落。

超过半数的企业在部署灾难恢复计划后并未加以充分测试。即使是那些确实组织过测试的企业也会在灾难恢复工作的人员、流程及必要工具等领域平均犯下五种错误。

灾难恢复既不令人振奋也非易于现实，但却扮演着决定企业命运的重要角色，他补充称。

“成功保护业务流程未必能让管理者平步青云，但一旦发生严重事故，我们很可能立刻被扫地出门。”

故事寓意：上阵之前最好先试试自己的武器灵不灵光。

致命IT失误七：向权贵进言

十年之前，Bob（化名）为一家在全国拥有超过一千家网点的发薪日贷款机构工作。（Bob本人要求在这个故事中隐瞒他的真实姓名）。他曾被任命为ASP系统的架构调整负责人，这套运行在各个网点本地服务器中的系统负责处理历史数据。但在着手工作之前，他必须首先通过将各网站中的几十套Web法律规程转换到数据库中来证明自己的技术实力。

一个周五的下午，也就是他接手这个工作岗位的半年之后（试用期结束的两周之前），IT副总裁在每周员工会议上提出了他对公司提出的五年远景规划。在副总裁两个小时的演讲中，Bob归纳出了四条要点：

- 拼搏到底
- 修复错误

◆ 保持良好现状

◆ 不要引入新技术

“这实在是把我难倒了，”Bob无奈地表示。“在我看来，‘如果按这套思路来搞，公司请我来干嘛？’他们每年要花数百万美元来维护由五十多位不同开发者所创建的网站——说实话，这网站简直是千疮百孔。”

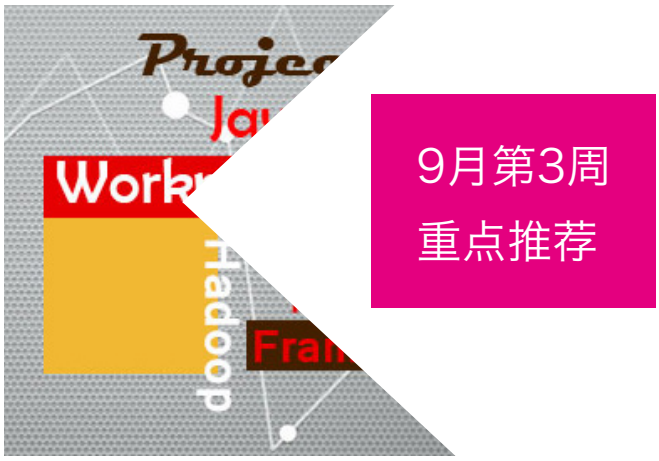
就在当天晚些时候，Bob走向副总裁的办公室并关上房门。

“他问我，‘你觉得我的观点怎么样？’”Bob告诉我们。“我说，‘坦率地说，先生，这简直相当于没有观点。您所表述的只是一套维护规划而非远景预期。’”

副总裁对他的坦率表示感谢，并称赞了他的勇气。然而当新的周一来临，Bob步入自己的办公室时，却发现自己已经丢掉了工作。

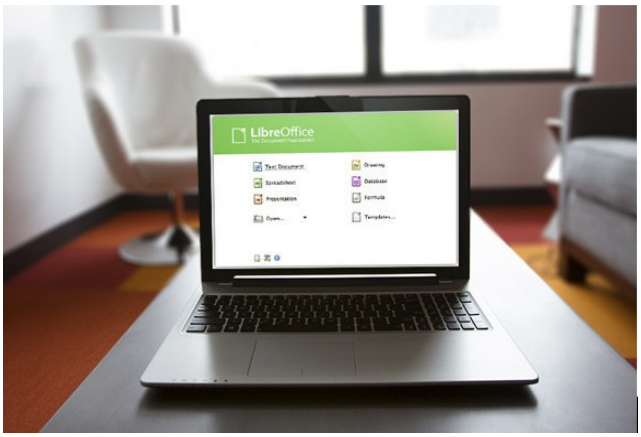
“我边吹口哨边开车回家，”他表示。“我从来没像现在这样为自己的失业感到庆幸。我决定永远不再把自己的职业前途押在金玉其外的混蛋身上。这次经历之后，我决定创立自己的事业、并一直奋斗至今。”■

原文链接：
http://developer.51cto.com/art/201309/410881.htm



为了帮助大家指明一条正路，我们将汇总一系列用于取代目前中小企业必备商用软件的开源方案。

商业软件已成过去：十款为小型企业打造的开源替代品



没有技术作为依托，我们根本无法让一家企业——即使是小型企业——正常运转。在如今的日常工作中，我们需要使用计算机、智能手机、文件存储、网站以及一整套用于托管其它技术资产的主机。近年来经济萎靡、预算紧缩，我们要如何利用有限的资金打理好技术工作？

当然，某些成本支出根本无法避免，但的确有一些工具用不着大家掏出信用卡就能合理合法地加以使用。当下，最简单的资金节约方式就是利用开源替代方案取代原本昂贵的商业软件。开源社区提供一系列解决方案、附带专业级功能，但同时又不会贴出令人咋舌的价格标签。

为了帮助大家指明一条正路，我们将汇总一系列用于取代目前中小企业必备商用软件的开源方案。如果大家能够按照本文的指示采纳开源产品，那么每位用户将能节约近两千美元的支出。兴奋了？激动了？别急，下面咱们慢慢聊。

Office套件: LibreOffice

在文字处理、电子表格及演示软件三大支柱的推动下，微软Office已经成为绝大多数企业在生产领域的核心与灵魂。不过Office 2013家用及企业版本将给每位用户带来220美元的使用成本，Office 365订阅价格更达到每位用户每年150美元。

LibreOffice与微软Office文件格式相兼容。

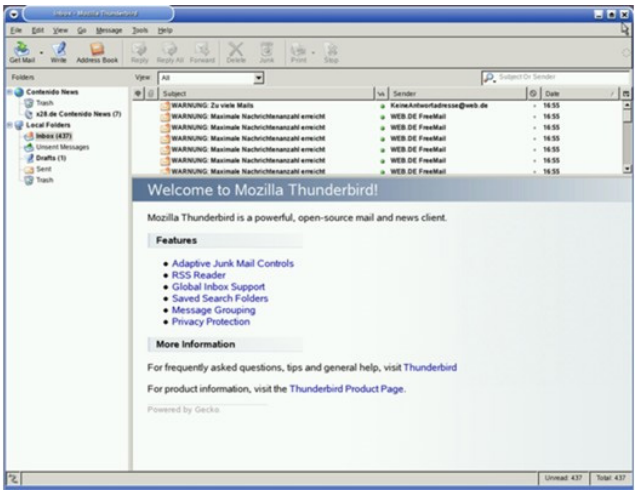
LibreOffice通过一套免费软件包带来与Office几乎相同的常用功能。它的工作原理与标准微软Office文件格式相同，所以我们仍然能够打开或者查看其他人发来的Office文件，或者把自己的LibreOffice文档分享给使用微软Office的合作伙伴或

者客户。它还集成了内容管理系统与在线文档存储机制，从而进一步简化协作流程。

官方网站：http://www.pcworld.com/article/2042552/review-libreoffice-4-liberates-you-from-microsoft-office.html

电子邮件: Thunderbird

对于大多数企业而言，电子邮件已经成为最重要的沟通方式。目前市场上可供选择的付费及免费电子邮件客户端可谓琳琅满目，但微软Outlook的使用范围最为广泛。Outlook属于微软Office家用与企业软件包的组成部分，当然在售价最高的微软Office专业版套件中也有提供。值得一提的是，大家还可以花上95美元单独购置这款方案。



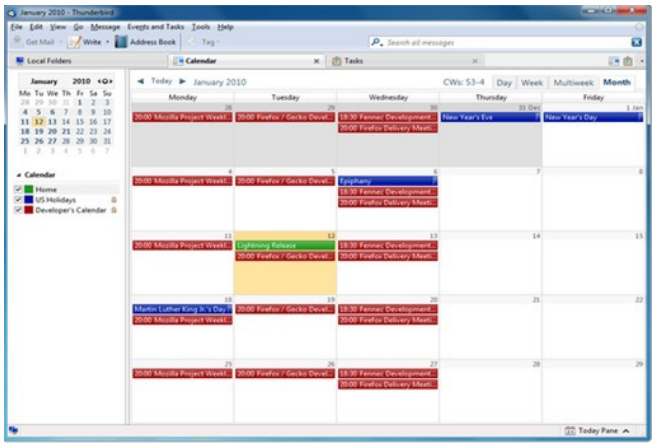
Thunderbird Filelink允许用户向邮件中将超大型文件附传上传至在线存储供应商处，并与收件人分享文件链接。

只要利用Thunderbird替代Outlook，大家就能为每位用户省下95美元的电子邮件开销。该产品由Mozilla公司开发——也就是火狐网络浏览器的创造者——Thunderbird提供非常全面的功能，包括标签式电子邮件、集成化聊天、智能文件夹以及网络钓鱼保护等。而且与火狐一样，它也可以通过插件进行定制。

官方网站：http://www.pcworld.com/product/947518/thunderbird.html

日程表: Lightning

大多数企业之所以对微软Outlook如此青睐，其日程表功能可谓功不可没。对于我们这些处于事业上升期的成功人士，约会、电话会议、销售会议以及项目截止日期等内容必须认真打理，这就需要大家利用一套强大的日程表工具来打理日常规划。



Mozilla公司的Lightning能够将与邮件有关的日程表任务集成至Thunderbird当中。

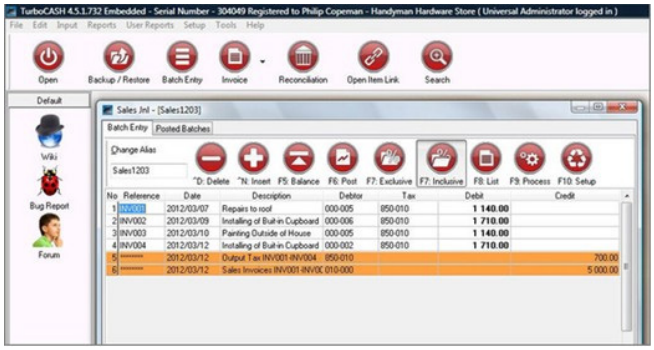
Mozilla带来另一款用于满足用户日常规划需求的免费工具。Lightning能够与Thunderbird相整合，从而管理我们的规划、发送并接收会议邀请、管理活动与任务等。大家还可以利用插件实现功能扩展。

官方网站：http://www.mozilla.org/projects/calendar/

财会事务: TurboCASH

许多企业依赖Quickbook来打理自己的财会事

务，这已经成为一种常态。Intuit公司的这款软件帮助用户管理报价与提议、发票、应付账款、应收账款等等，而且一切内容都显示在一套直观的界面当中。不过要使用QuickBook，大家首先要掏出150美元现金。



TurboCASH帮助大家追踪进入与流出企业的每一笔资金。

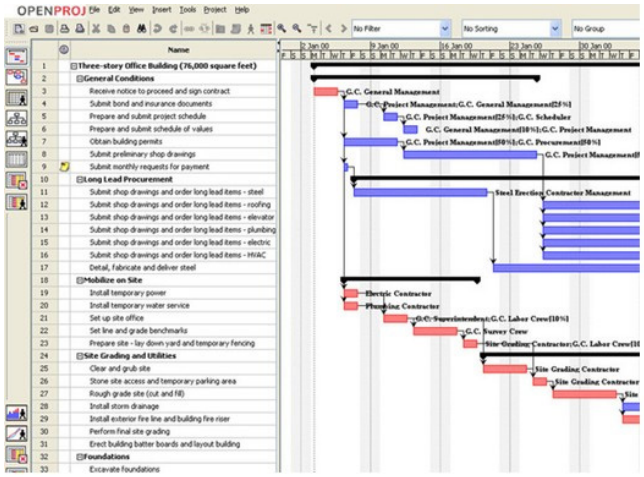
TurboCASH以免费方式为我们带来同样的功能。大家可以管理应收账款、应付账款、发票并进行银行对账等等。TurboCASH还拥有全面的报告功能，可以根据货币类型及产业领域的不同进行设定，从而满足世界各地小型企业的需求。

官方网站：http://turbocash.net/

项目管理: OpenProj

要保持项目始终运行于正轨之上，我们需要做出一系列努力。大家不仅要管理并分配人员、预算、资源以及其它必要因素，同时也得监控项目进度并关注截止日期。微软Project是一款出色的项目管理工具，但每位用户都会给企业带来456美元的沉重负担——对于中小型企业来说，这样的价位实在有些难以承受。

OpenProj支持多种文件格式，其中包括微软Project。

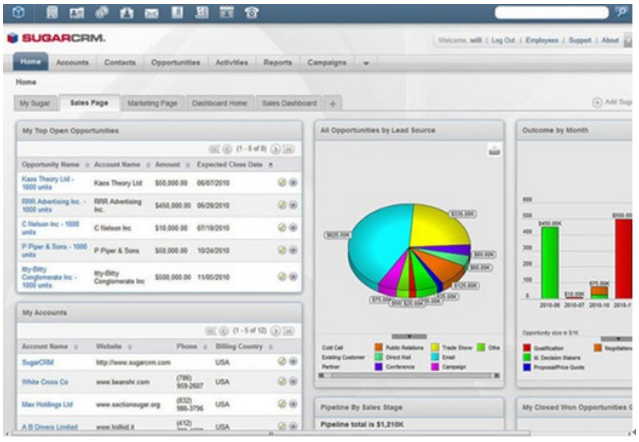


利用OpenProj作为替代方案能帮企业省下成吨的资金。它为用户带来非常相近的功能与处理能力，包括甘特图与计划评审技术图、工作分解结构、资源分解结构等等。更贴心的是，OpenProj还与微软Project一样提供了非常和缓的学习曲线。

官方网站：http://www.serena.com/index.php/en/products/openproj/

CRM（客户关系管理）: SugarCRM

要想让企业健康稳定步入发展轨道，时刻追踪前景与商机并通过工具管理客户关系正是不可或缺的前提。Salesforce.com公司已经在这一领域确立了自己的领导地位，但他们的产品要求每位用户每年支付300美元。



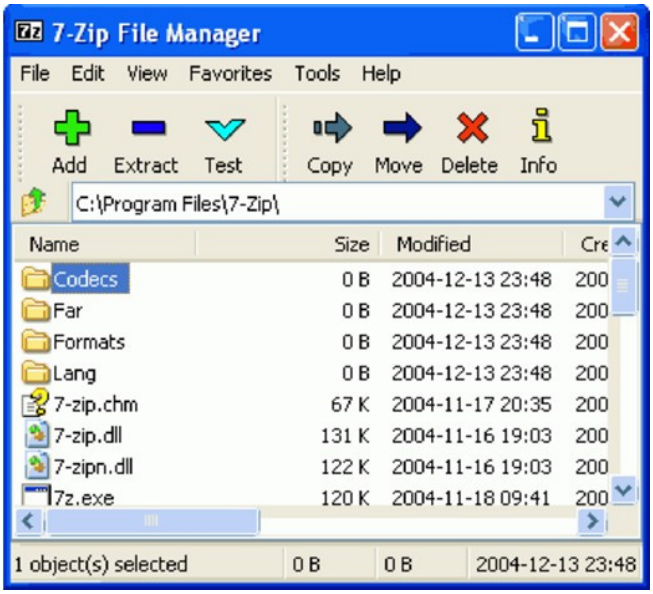
SugarCRM提供了多个版本，这样用户就能随着业务增长实现功能扩展。

SugarCRM是一款全功能开源平台，而且提供与Salesforce.com方案类似的功能。SugarCRM社区版完全免费，而随着业务及实际需求的不断增长，大家可以逐步选择专业版、公司版、企业版或者旗舰版等付费版本。无论大家使用的是哪个版本，都可以自由访问产品源代码，从而根据实际需求对CRM工具做出调整。

官方网站：http://www.sugarcrm.com/community

文件归档: 7-Zip

WinZip实际上已经成为一套标准化方案，提供一系列强大的工具及选项，能够将文件压缩为不同格式并轻松实现解压。虽然价格还不至于给预算带来太大压力，但30美元的成本仍然不容忽视。



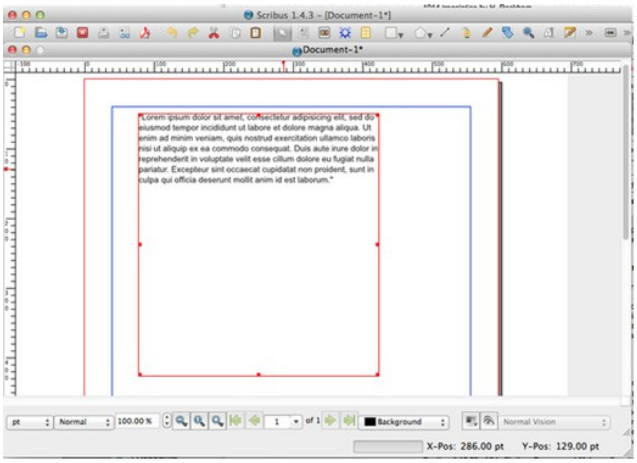
7-Zip提供的文件归档功能搭配256位AES加密机制。

作为替代方案，大家不妨认真考虑7-Zip。它的工作原理与广泛的压缩格式支持能力都与WinZip并无二致。它还提供256位AES加密机制，与Windows集成且提供包含79种语言的完善本地化效果。

官方网站：http://www.pcworld.com/article/232451/7zip_64bit_version.html

桌面出版: Scribus

不少中小型企业都会创建自己的营销方案与广告样本、设计宣传册、宣传单以及其它内容，这就到了微软Publisher大显身手的时候。与Outlook类似，Publisher同样包含在某些价格较高的微软Office版本当中，大家也可以花上95美元单独进行购买。



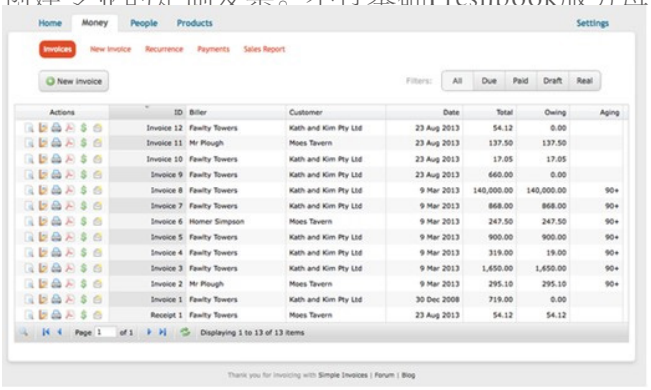
Scribus给大家带来创建专业级营销材料所必需的全部工具。

Scribus能给我们带来相同的页面布局能力。这款开源软件中包含我们在创建专业级营销材料时所必需的全部工具，包括利用分色进行press-ready输出、CMYK与专色、ICC颜色管理等。

官方网站：http://www.scribus.net/canvas/Scribus

发票: Simple Invoices

无论大家打理的是哪方面业务，其中最重要的功能之一（也许就是最重要的功能）在于收取资金。很多小型企业会利用Freshbook等服务为客户创建专业的定制发票。不过基础Freshbook服务每



Simple Invoices允许用户通过任何网络浏览器处理账单事务。

要找到一种经验实惠的替代方案，大家不妨考虑Simple Invoices。这款发票工具允许用户追踪客户、管理重复计费、调整税率等。而且与Freshbook一样，大家可以通过任何网络浏览器对其发起访问。

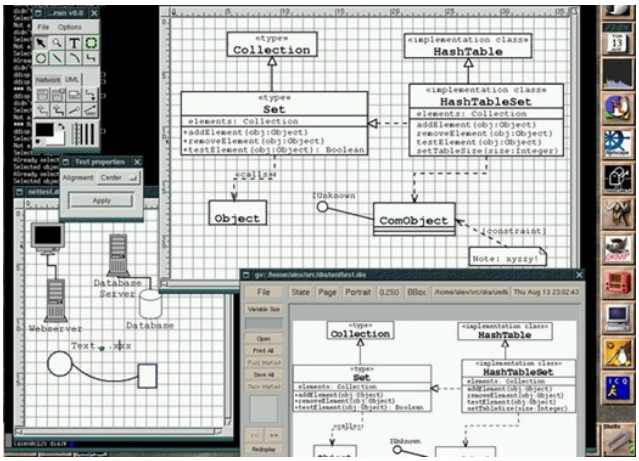
官方网站：http://www.simpleinvoices.org/

图表创建: Dia

如果大家需要创建流程图或者其它可视化图表，微软Visio是一款非常出色的生成工具。但不出所料，250美元每份授权的高昂成本令人望而却步。

Dia允许用户将复杂信息转化为可视化流程图及图表。

别担心，试试Dia。站在Visio巨人的肩膀上，Dia中包含着一系列工具以及特殊对象，旨在帮助客户创建实体关系图、流程图以及网络图等等。它还能够将统计结果保存为各种格式的图片文件，例如XML、EPS、WMF、SCG、PNG以及



官方网站：http://www.pcworld.com/article/232070/dia.html

采用开源解决方案

虽然开源工具本身完全免费，但大家应该明确意识到它也会带来一些隐性成本。无论各位打算从零开始直接采用开源方案、还是从现有软件中转移到开源系统当中，我们在享受理想的处理效果之前都必须面对新的学习曲线。如果从现有工具转移到新型开源工具，大家可能还需要一套专门的转换或者数据迁移计划。

此外，大多数开源项目都拥有强大的支持者阵营以及分享知识的技术社区。不过一旦遇上不顺心的麻烦，大家没办法直接打电话给供应商抱怨自己的遭遇。某些开源项目或者第三方IT企业会针对开源工具提供收费服务支持，但这样一来就违背了选择开源的首要目的——省钱，不是吗？ ■

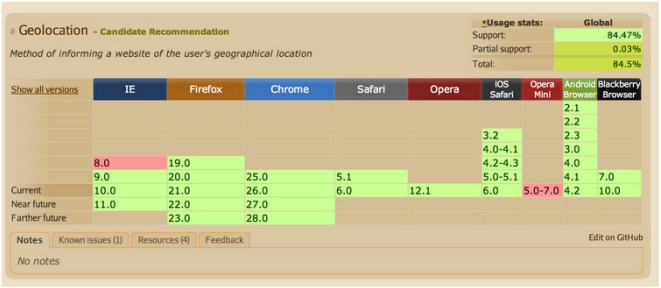
原文链接：
http://developer.51cto.com/art/201308/408687.htm

如何利用HTML5与MongoDB创建位置感知Web应用程序

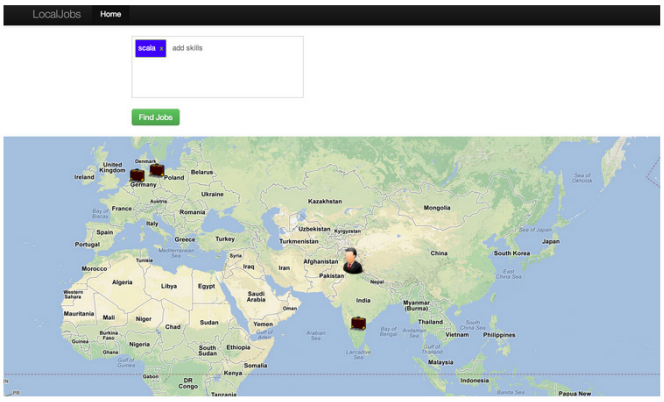
□ 核子可乐译

在日常生活中，我们都离不开位置识别类应用程序。Foursquare、Facebook等应用程序帮助我们和我们的家人朋友分享当前位置（或者正在参观的景点）。而像Google Local这样的应用则帮助我们找到当前位置附近有哪些自己需要的服务设施或业务场所。如此，如果我们需要找到一家离自己最近的咖啡厅，完全可以通过Google Local快速获取建议并立刻动身前往。这不仅大大方便了日常生活，还能够帮助企业将自己的产品推销给更理想的受众群体。无论是对消费者还是对企业，这都堪称完美的双赢局面。

要创建这样一款应用程序，大家首先需要获取用户的地理位置信息。根据维基百科的解释，“地理信息是指某个对象所处的现实地理位置”。就目前来看，Web应用程序中还没有出现标准化的用户地理位置获取方式。虽然Google Gears这样的开源库能够从用户处获取位置信息，但这套库已经停止发展、只能运行在旧版本浏览器当中而且不支持W3C地理位置API。W3C GeoLocation API提供了一套规范，能够通过标准化脚本访问与托管设备相关的地理信息。Geo Location并不提供对HTML 5的官方支持，但这仍然无法阻止人们的热情，而且我们经常听说开发人员将GeoLocation API与HTML 5相对接。该API以用户所收集的地理信息为基础建立抽象层，从而保证所有浏览器都支持地理定位API。大家可以访问http://caniuse.com/#feat=geolocation获取下列图表。

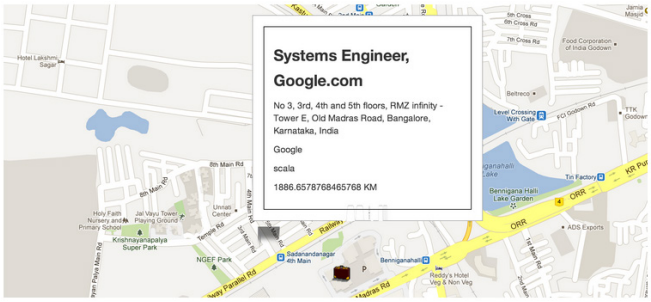


在本文中，我们将创建一款能够感知地理位置的找工作应用。应用程序将根据用户的特定技能（例如Java、Scala以及MongoDB等）寻找最近的求职地点。应用将利用W3C GeoLocation API实现用户定位。接下来，应用程序将用户位置绘制在谷歌地图当中。大家可以访问http://localjobshtml5-cix.rhcloud.com/获取这款应用。用户图标对应用户当前地理位置，公文包图标则对应目标求职地点。



如果大家点击任何公文包图标，地图会如下图所示自动放大。而当我们关闭信息窗口，画面会再次缩小。另外，大家可以在标记中查看求职场所与当前位置之间的距离、对应职务以及其它相关资

料。用户位置与工作位置之间的距离由MongDB的地理空间功能所支持，我们会在后面的文章中进一步讨论这个话题。



应用程序技术堆栈

这款应用的创建需要使用以下技术堆栈：

Java EE 6：我们将使用数项Java EE 6规范——JAX-RS以及CDI。JAX-RS属于针对Restful Web服务的Java API，其作用在于根据REST架构模式为网络服务创建提供Java API。CDI则是Context and Dependency Injection（背景与关联性注入）的缩写。CDI允许开发者将Java EE组件与生命周期背景进行绑定、注入，而后通过事件触发与观察机制以松散的耦合方式实现交互。

MongoDB：MongoDB是一套面向文档的NoSQL数据存储机制。我们将把工作数据保存在MongoDB当中并在应用程序中使用其地理空间功能。

HTML 5：我们将利用HTML 5创建应用程序客户端，并利用W3C GeoLocation API获取用户的当前位置。

谷歌地图：应用程序将利用谷歌地图来处理用户位置以及求职信息。

OpenShift：应用程序将被部署到OpenShift公共PaaS当中。

应用程序源代码

这款应用程序的源代码被发布在GitHub当中，地址为：https://github.com/shekhargulati/localjobshtml5

前续条件

在我们着手创建应用程序之前，首先需要进行以下几项设置任务：

1. 注册一个OpenShift账户。账户注册完全免费，而且红帽将为每位用户免费提供三套Gear用于运行应用程序。截至本文截稿时，该账户可以获得1.5GB内存容量与3GB磁盘存储空间。

2. 在设备上安装rhc客户工具。rhc是一套ruby gem包，因此大家需要在设备上安装ruby 1.8.7或者更高版本。要安装rhc，大家需输入：

```
1. sudo gem install rhc
```

如果当前已经安装过ruby，请确保其处于最新版本。要更新rhc工具，请执行如下所示命令：

```
1. sudo gem update rhc
```

如需其它相关rhc命令行工具设置说明，请点击下列网址查看相关资料：https://openshift.redhat.com/community/developers/rhc-client-tools-install

1 利用rhc setup命令设置OpenShift账户。这条命令将帮助大家创建一个命名空间并将自己的ssh密钥上传至OpenShift服务器。

开始创建应用程序

现在我们已经完成了全部前续设置工作，现在开始创建应用程序。我们将从创建OpenShift应用程序开始。在与PaaS协作时，大家首先需要明确一点：PaaS是用来创建应用程序的。因此，现在我们要摆脱过去以虚拟机或者服务器为中心的理念，将

全部精力集中在应用程序身上。

创建JBossEAP MongoDB OpenShift应用程序

要创建名为“localjobs”且使用JBossEAP与MongDB的应用程序，我们首先要执行以下命令：

```
1. rhc app create localjobs jbosseap
   mongodb-2.2
```

这将为我们创建一套应用程序容器，也就是所谓gear，并为其配置全部必要的SELinux政策以及cgroup配置。OpenShift还将为我们设置一个私有git库，并将该库克隆到本地系统当中。最后，OpenShift会将DNS发送至外部环境。大家可以通过http://localjobs-domain-name.rhcloud.com访问该应用。将其中的域名替换为您自己的独特域名即可。

上述命令将创建一套标准化Maven项目模板。有趣的是，在pom.xml文件中存在一段名为openshift的配置信息，如下所示。因此，当大家将自己的源代码推送至OpenShift时，该Maven配置文件将付诸执行。该配置文件不会引发任何影响——而只是创建一个名为ROOT的war文件，从而保证我们的应用程序可用于root背景之下。

```
1. <profiles>
2. <profile>
3. <id>openshift</id>
4. <build>
5. <finalName>localjobs</finalName>
   <plugins>
6. <plugin>
7. <artifactId>maven-war-plugin</artifactId>
8. <version>2.1.1</version>
   <configuration>
9.  <outputDirectory>deployments</
   outputDirectory>
   <warName>ROOT</warName>
10.      </configuration>
11.      </plugin>
12.      </plugins>
13.      </build>
14.      </profile>
15.      </profiles>
```

接下来，我们将把index.html与snoop.jsp两个文件从自己的git库中移除——它们的历史使命已经完成。如果大家不太熟悉git的运作方式，请点击此处阅读由Lars Vogel撰写的上手指南。

```
1. git rm -f src/main/webapp/index.html src/
   main/webapp/snoop.jsp
2. git commit -am  “deleted template files”
```

添加MongoDB Java驱动程序关联性

由OpenShift创建的pom.xml文件已经拥有全部与Java EE 6相关的关联性。为了使用MongoDB，我们还需要添加MongoDB Java驱动关联性。我使用的是MongoDB Java驱动的最新版本。将下列关联性内容添加到pom.xml文件当中。大家可以点击此处在github上查看完整的pom.xml文件。

```
1. <dependency>
2.      <groupId>org.mongodb</groupId>
3. <artifactId>mongo-java-driver</artifactId>
4. <version>2.10.1</version>
5. </dependency>
```

启用CDI

CDI代表背景与关联性注入。之所以在应用程序

中使用CDI，是因为我们需要利用关联性注入来代替手动创建对象。CDI容器将管理bean生命周期，这样我们作为开发者只需要编写业务逻辑即可。为了让JBossEAP应用程序服务器了解到我们正在使用CDI，我们需要在WEB-INF文件夹下创建一个beans.xml文件。该文件可以保持空白，但它的存在会使容器了解到需要加载CDI框架。Beans.xml文件的内容如下所示：

```
1. <?xml version=”1.0”?>
2. <beans xmlns=”http://java.sun.com/xml/ns/
   javaee”
3. xmlns:xsi=”http://www.w3.org/2001/
   XMLSchema-instance”
   xsi:schemaLocation=”http://java.sun.com/
   xml/ns/javaee http://jboss.org/schema/cdi/
   beans_1_0.xsd”/>
```

编写MongDB数据库连接类

接下来，我们将创建一个应用程序作用域bean，用于管理MongoDB数据库连接。该连接类同时起效于本地系统与OpenShift端。大家可以点击此处在github中查看该类的完整内容。

（代码略，见原文）

在应用程序运行过程中，@ApplicationScoped bean将始终存在，并在应用程序关闭的同时被删除。这正是我们希望通过MongoDB驱动所达到的连接池对象保留效果。

编写RESTful后端

现在我们开始利用JAX-RS为自己的应用程序编写RESTful后端。我们将通过创建一个用于扩展javax.ws.rs.ApplicationPath的类激活JAX-RS。大家需要指定一条基础url，并将其作为网络服务的访问地址。要实现这一目的，我们需要利用ApplicationPath注释为这个类添加注释。如下列代码所示，我利用“/api”作为基础URL：

```
1. import javax.ws.rs.ApplicationPath;
2. import javax.ws.rs.core.Application;
3. @ApplicationPath(“/api”)
4. public class JaxRsActivator extends Application
   {
5. /* class body intentionally left blank */
6. }
```

在成功激活了JAX-RS之后，我们现在可以编写自己的REST服务。大家可以访问http://localjobs-domain-name/api/jobs/{skills}?longitude={longitude}&latitude={latitude}以查看REST端点。该REST端点将搜寻周边经纬度范围内全部与求职者技能相符的工作岗位。

（代码略，见原文）

上面所示的代码会创建一条MongoDB附近位置查询，其结果文件数量被限制为10个。MongoDB返回的结果将作为数据中的数值。由于我们利用经度与纬度进行定位，返回的数据也以经纬度为基础。不过MongoDB还提供一套距离换数选项，允许我们将经纬度结果换算成更易理解的公里或者英里。在上面的代码中，我将经纬度结果转换为111公里。最后，我们将数据转换为一个名为Job的域对象并将其返回。@Produces注释将负责将数据转换至JSON。

将数据载入至MongoDB当中

执行下列命令将数据载入至运行在OpenShift gear中的MongoDB。

在本地设备上，运行rhc app show。这条命令将返回应用程序的详细信息，如下所示：

（代码略，见原文）

记录下SSH URL并利用scp命令将jobs-data.json文件复制到我们的应用程序gear当中。大家可以点击此处下载jobs-data.json文

```
1. $ scp jobs-data.json <ssh url>:app-root/data
```

接着将SSH插入到应用当中，使用如下所示的rhc app ssh命令：

```
1. $ rhc app ssh -a localjobs
```

将ssh导入至应用程序gear中后，将目录变更为app-root/data，也就是我们复制jobs-data.json文件的目录。

```
1. $ cd app-root/data
```

下面运行mongoimport命令将数据导入至MongoDB数据库当中。

（代码略，见原文）

上面显示的代码将把159个job对象导入至MongoDB当中。

最后，我们需要在工作集合中创建地理位置索引。MongoDB只支持二维地理位置索引。大家只能为每个集合匹配一套地理位置索引。在默认情况下，二维地理位置索引假设经度与纬度数值在-180（含180）到180（不含180）之间（即[-180，180]）。要创建地理信息索引，需要执行下列命令：

```
1. $ mongo
2. $ use localjobs
3. $ db.jobs.ensureIndex({ “location” : “2d” })
```

测试RESTful服务

下面，我们将提供源代码并向OpenShift推送变更内容，即创建项目、创建新的war文件并将其部署到运行在OpenShift上的JBossEAP当中。

```
1. $ git add .
2. $ git commit -am  “RESful backend done”
3. $ git push
```

在代码创建与war文件部署工作完成后，我们就可以利用curl命令对REST服务进行测试了。

（代码略，见原文）

美化应用程序

现在我们已经证实了应用程序的REST服务工作正常，接下来要做的是构建应用的用户界面。在本文中，我们只需创建一套非常简单的应用用户界面，即提供一套表单，用户可以借助它输入个人技能，并通过div承载谷歌地图渲染完成的求职场所与用户位置。如下所示在src/main/webapp文件夹中创建一个index.html文件：

（代码略，见原文）

上面显示的index.html是一个HTML 5文件，而且使用HTML 5的文档类型。我们的应用使用Twitter Bootstrap，这是一款免费工具集合，用于创建网站以及web应用程序。它包含了以HTML以及CSS为基础的设计模板，提供全套排版、表格、按钮、图表、导航、其它界面组件以及备选JavaScript扩展。大家可以点击此处从本项目的github库中获取全部相关css.js文件。

检查GeoLocation支持

由于我们的应用程序以用户位置为基础，因此在进一步调整应用程序之前需要首先检查GeoLocation API。为了检查用户浏览器对GeoLocation API的支持效果，需要将如下所示记录

准备函数添加进来。如果用户浏览器支持GeoLocation，那么导航对象中将具有geolocation对象。大家还可以利用Modernizr等开源库检测HTML 5功能。如果用户浏览器不支持geolocation，大家需要禁用表单提交按钮。

（代码略，见原文）

在提交表单中查找工作

现在我们已经确认用户浏览器能够支持GeoLocation API，接下来要做的就是根据用户的个人技能为其查找理想工作。此项目利用Backbone.js为我们的客户端代码添加结构。如果大家对backbone.js不太熟悉，可以点击此处查看我之前发表的博文《利用Backbone.js、JaxRS、MongoDB以及OpenShift创建单页面Web应用程序》，那里提供了与利用backbone.js创建应用有关的详细说明。请将app.js文件考虑到src/main/webapp目录下的js文件夹当中。下面展示的是经过精简的app.js文件内容，这是为了适当缩减本文的篇幅。

（代码略，见原文）

下面我们一起来解读代码的具体含义。

1. 上面展示的代码旨在创建一个backbone路由实例，并将其作为root DOM的主div。下面我们点击基础url，路由机制会调用映射HomeView的showHomePage函数。渲染函数中的HomeView用于通过id map-canvas清空div。

2. 在HomeView当中，我们拥有一套针对表单提交的事件侦听器。因此，当用户输入个人技能并按下“提交”按钮后，findJobs函数将被调用。

3. findJobs函数是一切运行的基础。

3.1 我们首先利用技能名称获取输入值，然后利用逗号将内容分割，这样就构成了一套技能数组。

3.2 我们接着创建一个谷歌地图对象并为其设置一些默认值。

3.3 下面我们调用navigator.geolocation对象上的getCurrentPosition方法。此方法只有一项必要参数success_callback与两项可选参数error_callback，外加可选对象PositionOptions。

3.4 如果getCurrentPosition被调用成功，则继续调用success_callback。这条回调函数拥有一项参数——position。这个position对象负责保留用户的经伟度结果，并在地图上绘制用户的当前位置。

3.5 在用户位置绘制完成之后，则通过jQuery进行获取调用。

3.6 最后所有结果都将经过迭代并显示在地图之上。

推送代码

现在大家可以将代码推送至OpenShift处并查看应用程序在云中的运行效果。

（代码略，见原文）

按照我所罗列的提示内容，应用程序将运行在https://localjobs-domain-name.rhcloud.com/位置。大家可以将具体域名替换为自己的命名空间。

总结

在本篇博文中，我们共同探讨了如何利用HTML 5 GeoLocation API以及MongoDB Geo位置索引功能创建位置感知类应用程序。希望文章内容能给大家的开发工作带来启示，感谢阅读。■

原文链接：
http://developer.51cto.com/art/201308/408685.htm

Servlet3中异步Servlet特性介绍

■ 在Jave EE 6规范中,关于Servlet 3规范的相关功能增强,一直是让大部分用户忽略的,连直到最新的Spring MVC 3.2才支持Servlet 3的异步调用。这可能跟大部分用户使用的JAVE EE容器依然是旧的有关系。

在Jave EE 6规范中,关于Servlet 3规范的相关功能增强,一直是让大部分用户忽略的,连直到最新的Spring MVC 3.2才支持Servlet 3的异步调用。这可能跟大部分用户使用的JAVE EE容器依然是旧的有关系（如支持Servlet 3规范的需要Tomcat 7，但目前不少用户还在使用Tomcat 6）。

在本文中，将以实际的例子来讲解下Servlet 3规范中对异步操作的支持。

首先要简单了解，在Servlet 3中，已经支持使用注解的方式去进行Servlet的配置，这样就不需要在web.xml中进行传统的xml的配置了，最常用的注解是使用@WebServlet、@WebFilter、@WebInitParam,它们分别等价于传统xml配置中的<Servlet>、<WebFilter>、<InitParam>，其他参数可参考Servlet 3中的规范说明。

```
1. @WebServlet(“/LongRunningServlet”)
2. public class LongRunningServlet extends
   HttpServlet {
3. private static final long serialVersionUID = 1L;
4. protected void doGet(HttpServletRequest
   request,
5. HttpServletResponse response) throws
   ServletException, IOException {
6. long startTime = System.currentTimeMillis();
7. System.out.println(“LongRunningServlet
   Start::Name=”
8. + Thread.currentThread().getName() + “::ID=”
9. + Thread.currentThread().getId());
10.     String time = request.
   getParameter(“time”);
11.     int secs = Integer.valueOf(time);
12.     //如果超过10秒，默认用10秒
13.     if (secs > 10000)
14.     secs = 10000;
15.     longProcessing(secs);
16.     PrintWriter out = response.getWriter();
17.     long endTime = System.
   currentTimeMillis();
18.     out.write(“Processing done for “ + secs
   + “ milliseconds!!”);
19.     System.out.println(“LongRunningServlet
   Start::Name=”
20.     + Thread.currentThread().getName() +
   “::ID=”
21.     + Thread.currentThread().getId() +
   “::Time Taken=”
22.     + (endTime - startTime) + “ ms.”);
23.     }
24.     private void longProcessing(int secs) {
```

```
25.     //故意让线程睡眠
26.     try {
27.     Thread.sleep(secs);
28.     } catch (InterruptedException e) {
29.     e.printStackTrace();
30.     }
31.     }
32.     }
```

下面我们开始了解下，如果不使用异步特性的一个例子，代码如下：

运行上面的例子，输入

http://localhost:8080/AsyncServletExample/LongRunningServlet?time=8000，则可以看到输出为：

LongRunningServlet Start::Name=http-bio-8080-exec-34::ID=103

1. LongRunningServlet Start::Name=http-bio-8080-exec-34::ID=103::Time Taken=8002 ms.

可以观察到，在主线程启动后，servlet线程为了处理longProcessing的请求，足足等待了8秒，最后才输出结果进行响应，这样对于高并发的应用来说这是很大的瓶颈，因为必须要同步等到待处理的方法完成后，Servlet容器中的线程才能继续接收其他请求，在此之前，Servlet线程一直处于阻塞状态。

在Servlet 3.0规范前，是有一些相关的解决方案的，比如常见的就是使用一个单独的工作线程(worker thread)去处理这些耗费时间的工作，而Servlet 容器中的线程在把工作交给工作线程处理后则马上回收到Servlet容器中去。比如Tomcat的Comet、WebLogic的的FutureResponseServlet和WebSphere的Asynchronous Request Dispatcher都是这类型的解决方案。

但只这些方案的弊端是没办法很容易地在不修改代码的情况下迁移到其他Servlet容器中，这就是Servlet 3中要定义异步Servlet的原因所在。

下面我们通过例子来说明异步Servlet的实现方法：

1、 首先设置servlet要支持异步属性，这个只需要设置asyncSupported属性为true就可以了。

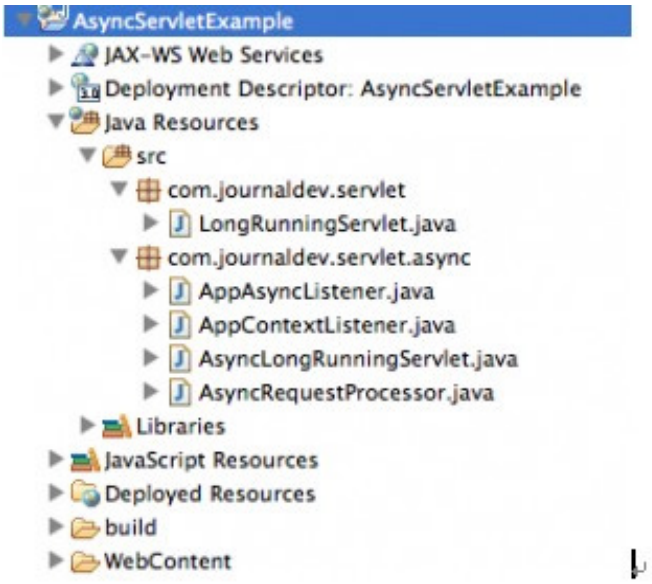
2、 因为实际上的工作是委托给另外的线程的，我们应该实现一个线程池，这个可以通过使用Executors框架去实现（具体参考http://www.journaldev.com/1069/java-thread-pool-example-using-executors-and-threadpoolexecutor一文），并且使用Servlet Context listener去初始化线程池。

3、 我们需要通过ServletRequest.startAsync()方法获得AsyncContext的实例。AsyncContext提供了方法去获得ServletRequest和ServletResponse的对象引用。它也能使用dispatch()方法去将请求forward到其他资源。

4、 我们将实现Runnable接口，并且在其实现方法中处理各种耗时的任务，然后使用AsyncContext对象去将请求dispatch到其他资源中去或者使用ServletResponse对象输出。一旦处理完毕，将调用AsyncContext.complete()方法去让容器知道异步处理已经结束。

5、 我们还可以在AsyncContext对象增加AsyncListener的实现类以实现相关的徽调方法，可以使用这个去提供将错误信息返回给用户（如超时或其他出错信息），也可以做一些资源清理的工作。

我们来看下完成后例子的工程结构图如下：



下面我们看下实现了ServletContextListener类的监听类代码：

```
1. AppContextListener.java
2. package com.journaldev.servlet.async;
3. import java.util.concurrent.
   ArrayBlockingQueue;
4. import java.util.concurrent.
   ThreadPoolExecutor;
5. import java.util.concurrent.TimeUnit;
6. import javax.servlet.ServletContextEvent;
7. import javax.servlet.
   ServletContextListener;
8. import javax.servlet.annotation.
   WebListener;
9. @WebListener
10.     public class AppContextListener
   implements ServletContextListener {
11.     public void contextInitialized(ServletContex
   tEvent servletContextEvent) {
12.                     // 创建线程池
13.                     ThreadPoolExecutor
   executor = new ThreadPoolExecutor(100, 200,
   50000L,
14.     TimeUnit.MILLISECONDS, new ArrayBlock
   ingQueue<Runnable>(100));
15.     servletContextEvent.getServletContext().
   setAttribute( “executor” ,
16.     executor);
17.     }
18.     public void contextDestroyed(ServletCont
   extEvent servletContextEvent) {
19.     ThreadPoolExecutor executor =
   (ThreadPoolExecutor) servletContextEvent
                   .getServletContext().
   getAttribute( “executor” );
20.     executor.shutdown();
21.     }
22.     }
```

然后是worker线程的实现代码，如下：

```
1. AsyncRequestProcessor.java
2. package com.journaldev.servlet.async;
3. import java.io.IOException;
4. import java.io.PrintWriter;
5. import javax.servlet.AsyncContext;
6. public class AsyncRequestProcessor
   implements Runnable {
7. private AsyncContext asyncContext;
8. private int secs;
9. public AsyncRequestProcessor() {
10.     }
```

```
11.  public AsyncRequestProcessor(AsyncCont
  ext asyncCtx, int secs) {
12.  this.asyncContext = asyncCtx;
13.  this.secs = secs;
14.  }
15.  @Override
16.  public void run() {
17.  System.out.println(“Async Supported? “
18.  + asyncContext.getRequest().
  isAsyncSupported());
19.  longProcessing(secs);
20.  try {
21.  PrintWriter out = asyncContext.
  getResponse().getWriter();
22.  out.write(“Processing done for “ + secs
  + “ milliseconds!!”);
23.  } catch (IOException e) {
24.  e.printStackTrace();
25.  }
26.  //完成异步线程处理
27.  asyncContext.complete();
28.  }
29.  private void longProcessing(int secs) {
30.  // 休眠指定的时间
31.  try {
32.  Thread.sleep(secs);
33.  } catch (InterruptedException e) {
34.  e.printStackTrace();
35.  }
36.  }
37.  }
```

请在这里注意AsyncContext的使用方法，以及当完成异步调用时必须调用asyncContext.complete()方法。

现在看下AsyncListener类的实现

```
1. package com.journaldev.servlet.async;
2. import java.io.IOException;
3. import java.io.PrintWriter;
4. import javax.servlet.AsyncEvent;
5. import javax.servlet.AsyncListener;
6. import javax.servlet.ServletResponse;
7. import javax.servlet.annotation.WebListener;
8. @WebListener
9. public class AppAsyncListener implements
  AsyncListener {
10.  @Override
11.  public void onComplete(AsyncEvent
  asyncEvent) throws IOException {
12.  System.out.println(“AppAsyncListener
  onComplete”);
13.  // 在这里可以做一些资源清理工作
14.  }
15.  @Override
16.  public void onError(AsyncEvent
  asyncEvent) throws IOException {
17.  System.out.println(“AppAsyncListener
  onError”);
18.  //这里可以抛出错误信息
19.  }
20.  @Override
```

```
21.  public void onStartAsync(AsyncEvent
     asyncEvent) throws IOException {
22.  System.out.println(“AppAsyncListener
     onStartAsync”);
23.  //可以记录相关日志
24.  }
25.  @Override
26.  public void onTimeout(AsyncEvent
     asyncEvent) throws IOException {
27.  System.out.println(“AppAsyncListener
     onTimeout”);
28.  ServletResponse response = asyncEvent.
     getAsyncContext().getResponse();
29.  PrintWriter out = response.getWriter();
30.  out.write(“TimeOut Error in Processing”);
31.  }
32.  }
```

其中请注意可以监听onTimeout事件的使用，可以有效地返回给用户端出错的信息。最后来重新改写下前文提到的测试Servlet的代码如下：

```
33.  AsyncLongRunningServlet.java
34.  package com.journaldev.servlet.async;
35.  import java.io.IOException;
36.  import java.util.concurrent.
     ThreadPoolExecutor;
37.  import javax.servlet.AsyncContext;
38.  import javax.servlet.ServletException;
39.  import javax.servlet.annotation.WebServlet;
40.  import javax.servlet.http.HttpServlet;
41.  import javax.servlet.http.
     HttpServletRequest;
42.  import javax.servlet.http.
     HttpServletResponse;
43.  @WebServlet(urlPatterns = “/
     AsyncLongRunningServlet”, asyncSupported
     = true)
44.  public class AsyncLongRunningServlet
     extends HttpServlet {
45.  private static final long serialVersionUID
     = 1L;
46.  protected void doGet(HttpServletRequest
     request,
47.  HttpServletResponse response) throws
     ServletException, IOException {
48.  long startTime = System.
     currentTimeMillis();
49.  System.out.println(“AsyncLongRunningS
     ervlet Start::Name=”
50.  + Thread.currentThread().getName() +
     “::ID=”
51.  + Thread.currentThread().getId());
52.  request.setAttribute(“org.apache.
     catalina.ASYNC_SUPPORTED”, true);
53.  String time = request.
     getParameter(“time”);
54.  int secs = Integer.valueOf(time);
55.  // 如果超过10秒则设置为10秒
56.  if (secs > 10000)
57.  secs = 10000;
58.  AsyncContext asyncCtx = request.
     startAsync();
59.  asyncCtx.addListener(new
```

```
    AppAsyncListener());
60.     asyncCtx.setTimeout(9000);
61.     ThreadPoolExecutor executor =
    (ThreadPoolExecutor) request
62.     .getServletContext().
    getAttribute("executor");
63.     executor.execute(new
    AsyncRequestProcessor(asyncCtx, secs));
64.     long endTime = System.currentTimeMillis();
65.     System.out.println("AsyncLongRunningSe
    rvlet End::Name="
66.     + Thread.currentThread().getName() +
    "::ID="
67.     + Thread.currentThread().getId() + "::Time
    Taken="
68.     + (endTime - startTime) + " ms.");
69.     }
70.     }
```

下面运行这个Servlet程序，输入：

http://localhost:8080/AsyncServletExample/AsyncLongRunningServlet?time=8000,运行结果为:

```
AsyncLongRunningServlet Start::Name=http-
bio-8080-exec-50::ID=124
AsyncLongRunningServlet End::Name=http-
bio-8080-exec-50::ID=124::Time Taken=1 ms.
Async Supported? true
AppAsyncListener onComplete
```

但如果我们运行一个time=9999的输入,则运行结果为:

```
AsyncLongRunningServlet Start::Name=http-
bio-8080-exec-44::ID=117
AsyncLongRunningServlet End::Name=http-
bio-8080-exec-44::ID=117::Time Taken=1 ms.
Async Supported? true
AppAsyncListener onTimeout
AppAsyncListener onError
AppAsyncListener onComplete
Exception in thread "pool-5-thread-6" java.
lang.IllegalStateException: The request as-
sociated with the AsyncContext has already
completed processing.
        at org.apache.catalina.core.AsyncCon-
textImpl.check(AsyncContextImpl.java:439)
        at org.apache.catalina.core.AsyncCo-
ntextImpl.getResponse(AsyncContextImpl.
java:197)
        at com.journaldev.serv-
let.async.AsyncRequestProcessor.
run(AsyncRequestProcessor.java:27)
        at java.util.concurrent.
ThreadPoolExecutor$Worker.
runTask(ThreadPoolExecutor.java:895)
        at java.util.concurrent.
ThreadPoolExecutor$Worker.
run(ThreadPoolExecutor.java:918)
        at java.lang.Thread.run(Thread.
java:680)
```

可以看到,Servlet主线程很快执行完毕并且所有的处理额外的工作都是在另外一个线程中处理的,不存在阻塞问题。■

原文链接：
http://developer.51cto.com/art/201309/409211.htm

一步步教你如何用HTML 5拖拽功能打造购物车

■ 在本文中，将指导读者认识 HTML 5中的拖拽功能，并且使用它来打造一款基本的购物车。这个购物车很简单，我们只往其中放置一件商品然后检查是否已有商品了，如果已经存在同样的商品，则更 新其数量和价格。

在最新的HTML 5标准中,为最新的各类浏览器带来了拖拽功能。这意味着现在可以不通过其他框架如jQuery的辅助就能在页面上拖拽各种元素。在本文中，将指导读者认识 HTML 5中的拖拽功能，并且使用它来打造一款基本的购物车。这个购物车很简单，我们只往其中放置一件商品然后检查是否已有商品了，如果已经存在同样的商品，则更 新其数量和价格。本文要求读者有初步的HTML 5基础知识和一定基础的Javascript知识即可。

开始

首先我们需要设计购物车的基础结构和准备一系列的商品。为了简单演示使用了各类商品，这里只是使用HTML 5中的data_*属性（可参考https://developer.mozilla.org/en-US/docs/Web/Guide/HTML /Using_data_attributes），为各个商品添加了价格。简单来说，data_属性是让用户可以为某些标签存储一些额外的数据信息。代码如下所示：

```
1. <section id=”cart” class=”shopping-cart”>
2. <ul>
3. </ul>
4. <span class=”total”>0.00</span>
5. </section>
6. <section id=”products” class=”products”>
7. <ul>
8. <li id=”product-1” data-
   price=”2.00”><span>Product 1</span></li>
9. <li id=”product-2” data-
   price=”3.00”><span>Product 2</span></li>
10.  <li id=”product-3” data-
   price=”2.99”><span>Product 3</span></li>
11.  <li id=”product-4” data-
   price=”3.50”><span>Product 4</span></li>
12.  <li id=”product-5” data-
   price=”4.25”><span>Product 5</span></li>
13.  <li id=”product-6” data-
   price=”6.75”><span>Product 6</span></li>
14.  <li id=”product-7” data-
   price=”1.99”><span>Product 7</span></li>
15.  </ul>
16.  </section>
```

由于本文是主要讲解使用Javascript搭配HTML 5的新功能，因此不会使用大家常用的jQuery。下面编写addEvent事件，代码如下：

```
1. function addEvent(element, event, delegate ) {
2. if (typeof (window.event) != ‘undefined’ &&
   element.attachEvent)
3. element.attachEvent( ‘on’ + event, delegate);
4. else
5. element.addEventListener(event, delegate,
   false);
```

```
6. }
```

我们将使用的第一个事件是readystatechange 。该事件在当文档的状态发生改变的时候触发，我们期望将其状态设置为“complete”，直到附加上剩余的事件和逻辑代码。这个跟jQuery中的.ready事件类似。代码如下：

```
1. addEvent(document, ‘readystatechange’,
   function() {
2. if ( document.readyState !== “complete” )
3. return true;
4. });
```

对DOM查询

在HTML 5标准中，DOM的选择器方面也进行了更新，其中新增的功能中最有用的方法就是querySelectorAll。这个方法允许用户使用更复杂的css样式选择器（有点象jQuery）去查询页面元素。这比旧有的通过class和ID去进行查询简单些。

下面是其中一个例子：

```
1. var items = document.
   querySelectorAll(“section.products ul li”);
2. var cart = document.querySelectorAll(“#cart
   ul”)[0];
```

这里，首先是分别使用items和cart变量获得商品列表和购物车列表，留作稍候使用。可以看到，上面通过querySelectorAll的方法比以往更加方便了。既然已经有了产品的列表，那么我们将实现拖拽功能并且当产品被拖拽的时候，必须要有相关的事件进行监听。为了实现这个目的，将循环遍历商品列表并且使用setAttribute方法设置draggable属性为true，这里并且要为每个商品都添加dragstart事件。代码如下：

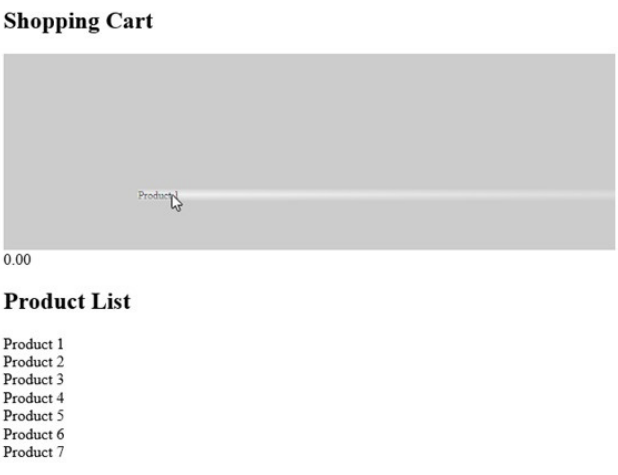
```
1. for (var i = 0; i < items.length; i++) {
2.     var item = items[i];
3.     item.setAttribute(“draggable”, “true”);
4.     addEvent(item, ‘dragstart’, onDrag);
5. };
```

在上面的代码中，请注意将dragstart方法委托定义为onDrag。这个方法的目的是设置拖拽的选项并且保存元素的id留作稍候通过使用dataTransfer.setData方法获得数据。dataTransfer事件对象允许我们指定默认的拖拽动作的效果。默认是既复制并移动，但这里我们强制令其为只移动元素。代码如下：

```
1. function onDrag(event){
2. event.dataTransfer.effectAllowed = “move”;
3. event.dataTransfer.dropEffect = “move”;
4. var target = event.target || event.srcElement;
5. var success = event.dataTransfer.
   setData(‘Text’, target.id);
6. }
```

设计购物车

我们先来看下设计的购物车大概是什么样子的，如下图：



看上去样子不大漂亮,但这个并不影响我们的示例教学用。可以看到用户可以拖拉在Product List中的商品到上面的购物车区域中。

默认元素是不接收drop事件的，因此为了能让某个元素能接收到拖拽的事件消息，我们要重写默认的行为。为了实现这个目的，使用了onDragOver方法。代码如下，所做的其实是阻止默认的dragover和dragenter事件行为：

```
1. function onDragOver(event){
2.     if(event.preventDefault) event.
   preventDefault();
3.     if (event.stopPropagation) event.
   stopPropagation();
4.     else event.cancelBubble = true;
5.     return false;
6. }
7. addEvent(cart, 'dragover', onDragOver);
```

接下来，我们要往购物车中增加商品了。只需要使用dataTransfer.getData方法从dataTransfer对象中取出id的值，有了id的值就可以从商品列表中找到商品。但要注意的是在往购物车中放商品前要检查该商品是否已经放置在里面了，

检查的方法很简单，只需要使用querySelectorAll方法就可以了，代码如下：

```
var exists = document.
querySelectorAll("#cart ul li[data-id='" +
id + "']");
```

接下来就很容易根据变量exists去判断是否购物车中已经存在商品，代码如下：

```
1. if(exists.length > 0){
2.     updateCartItem(exists[0]);
3. } else {
4.     addCartItem(item, id);
5. }
```

在上面的代码中,如果购物车中不包含任何商品,则调用下面的代码addCartItem。

```
1. function addCartItem(item, id) {
2.     var clone = item.cloneNode(true);
3.     clone.setAttribute('data-id', id);
4.     clone.setAttribute('data-quantity', 1);
5.     clone.removeAttribute('id');
6.     var fragment = document.
   createElement('span');
7.     fragment.setAttribute('class',
    'quantity');
8.     fragment.innerHTML = 'x 1';
9.     clone.appendChild(fragment);
10.         fragment = document.
   createElement('span');
11.         fragment.setAttribute('class',
    'sub-total');
12.         clone.appendChild(fragment);
13.         cart.appendChild(clone);
14.     }
```

如果购物车中不包含某件商品，则addCartItem方法中要做的事是克隆当前的商品项。现在可以指定的data-*的值。首先在克隆后的结点中，设置的是data-id结点，然后设置data-quantity属性为1并移除id属性（id属性在页面中是唯一的）。然后我们增加两个新的span到列表项中，这是用来显示小计项和产品的数量的。下图是从商品列表中拖拉到购物车中的情景：

更新购物车中的商品

如果一个商品已经在购物车中存在了，我们将要增加其数量，其中我们显示给用户的方式是单价*数量，因此，我们使用 getAttribute方法就可以获得当前的数量并且对其进行增加的操作，代码如下：

```
1. function updateCartItem(item){
2.     var quantity = item.getAttribute( 'data-quantity' );
3.     quantity = parseInt(quantity) + 1
4.     item.setAttribute( 'data-quantity' , quantity);
5.     var span = item.querySelectorAll( 'span.quantity' );
6.     span[0].innerHTML = ' x ' + quantity;
7. }
```

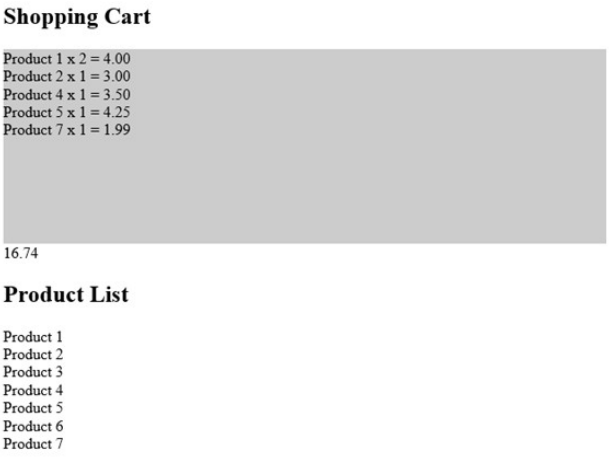
更新总价格

一旦购物车的商品数量增加了，我们就要重新计算总价格。这里我们再次使用了

querySelectorAll功能。我们只需要遍历购物车中的每一个商品并重新计算价格就可以了，这里是取了两位小数位。

```
1. function updateCart(){
2. var total = 0.0;
3. var cart_items = document.querySelectorAll( "#cart ul li" )
4. for (var i = 0; i < cart_items.length; i++) {
5. var cart_item = cart_items[i];
6. var quantity = cart_item.getAttribute( 'data-quantity' );
7. var price = cart_item.getAttribute( 'data-price' );
8. var sub_total = parseFloat(quantity * parseFloat(price));
9. cart_item.querySelectorAll( "span.sub-total" )[0].innerHTML = " = " + sub_total.toFixed(2);
10.     total += sub_total;
11.     }
12.     document.querySelectorAll( "#cart span.total" )[0].innerHTML = total.toFixed(2);
13.     }
```

从下图中可以看到当拖拽多个商品到购物车中，商品的总价格是会增加的：



最后我们总体看下所有的Javascript代码如下所示。

```
1. function addEvent(element, event, delegate ) {
2.     if (typeof (window.event) != 'undefined' )
3.         element.attachEvent( 'on' + event, delegate);
4.     else
5.         element.addEventListener(event, delegate, false);
6. }
7.
8. addEvent(document, 'readystatechange' , function() {
9.     if ( document.readyState !== "complete" )
10.             return true;
11.
12.         var items = document.querySelectorAll( "section.products ul li" );
13.         var cart = document.querySelectorAll( "#cart ul" )[0];
14.
15.         function updateCart(){
16.             var total = 0.0;
17.             var cart_items = document.querySelectorAll( "#cart ul li" )
18.             for (var i = 0; i < cart_items.length; i++) {
19.                 var cart_item = cart_items[i];
20.                 var quantity = cart_item.getAttribute( 'data-quantity' );
21.                 var price = cart_item.getAttribute( 'data-price' );
22.
23.                 var sub_total = parseFloat(quantity * parseFloat(price));
24.                 cart_item.querySelectorAll( "span.sub-total" )[0].innerHTML = " = " + sub_total.toFixed(2);
25.
26.                 total += sub_total;
27.             }
28.
29.             document.querySelectorAll( "#cart span.total" )[0].innerHTML = total.toFixed(2);
30.         }
31.
32.         function addCartItem(item, id) {
33.             var clone = item.cloneNode(true);
34.             clone.setAttribute( 'data-id' , id);
35.             clone.setAttribute( 'data-quantity' , 1);
36.             clone.removeAttribute( 'id' );
37.
38.             var fragment = document.createElement( 'span' );
39.             fragment.setAttribute( 'class' , 'quantity' );
40.             fragment.innerHTML = ' x 1' ;
41.             clone.appendChild(fragment);
```

```
42.
43.             fragment = document.
    createElement( ‘span’ );
44.             fragment.setAttribute( ‘class’ ,
    ‘sub-total’ );
45.             clone.appendChild(fragment);
46.             cart.appendChild(clone);
47.         }
48.
49.         function updateCartItem(item){
50.             var quantity = item.
    getAttribute( ‘data-quantity’ );
51.             quantity = parseInt(quantity) + 1
52.             item.setAttribute( ‘data-quantity’ ,
    quantity);
53.             var span = item.
    querySelectorAll( ‘span.quantity’ );
54.             span[0].innerHTML = ‘ x ‘ +
    quantity;
55.         }
56.
57.         function onDrop(event){
58.             if(event.preventDefault) event.
    preventDefault();
59.             if (event.stopPropagation) event.
    stopPropagation();
60.             else event.cancelBubble = true;
61.
62.             var id = event.dataTransfer.
    getData( “Text” );
63.             var item = document.
    getElementById(id);
64.
65.             var exists = document.
    querySelectorAll( “#cart ul li[data-id=’ ” + id
    + “’ ]” );
66.
67.             if(exists.length > 0){
68.                 updateCartItem(exists[0]);
69.             } else {
70.                 addCartItem(item, id);
71.             }
72.
73.             updateCart();
74.
75.             return false;
76.         }
77.
78.         function onDragOver(event){
79.             if(event.preventDefault) event.
    preventDefault();
80.             if (event.stopPropagation) event.
    stopPropagation();
81.             else event.cancelBubble = true;
82.             return false;
83.         }
84.
85.         addEvent(cart, ‘drop’ , onDrop);
86.         addEvent(cart, ‘dragover’ ,
    onDragOver);
87.
```

```
88.         function onDrag(event){
89.             event.dataTransfer.effectAllowed =
    “move”;
90.             event.dataTransfer.dropEffect =
    “move”;
91.             var target = event.target || event.
  srcElement;
92.             var success = event.dataTransfer.
  setData(‘Text’, target.id);
93.         }
94.
95.
96.         for (var i = 0; i < items.length; i++) {
97.             var item = items[i];
98.             item.setAttribute(“draggable”,
    “true”);
99.             addEvent(item, ‘dragstart’,
  onDrag);
100.        };
101.    });
```

本文的demo可以在http://developerdrive.developerdrive.netdna-cdn.com/wp-content/uploads/2013/09/cart.html中看到，完整代码可以在http://developerdrive.developerdrive.netdna-cdn.com/wp-content/uploads/2013/09/cart.zip中获得下载。■

原文链接：

http://developer.51cto.com/art/201309/410668.htm

链 接

银行业须直面“大数据”挑战

大数据时代，银行业面临着前所未有的信息安全挑战。银行存储着大量客户信息和敏感金融数据，一旦被非法窃取利用，将造成不可估量的损失。由于历史原因，国内银行所使用的服务器、存储、网络设备等核心IT基础设施,大部分是国外产品，即使有少数优秀的国产品牌，其核心芯片和协议软件也多源自国外。这就为外部势力攻击窃密提供了可能。并且现阶段网络攻击十分隐蔽，要检测和防范这些攻击行为非常困难。因此，银行业亟需进一步加强信息安全保障工作。

要建立健全立体协同的行业防控体系。一是加快制定大数据背景下的信息安全法律和法规，对相关犯罪行为加大惩治力度。二是建立信息安全防护的国家标准，大力引导企业进行实施和认证。三是加强行业监管，推动银行业信息安全管理工作向更深层次发展。四是深化同行业交流，实现同业的信息互通和知识共享。五是增强银行业务人员对大数据信息安全风险管控的责任感，切实规范操作。

要推进IT国产化，破除核心技术受制于人的困境。国产设备和国产软件生产商要不断提高产品质量和服务水平，银行也要支持国内优秀IT企业成长。

要创新技术防御理念，提升安全保障手段的可靠性。将风险关口前移，扩展安全分析深度和广度，将被动的事后分析转变为主动的事前防御。还可探索建立主动应对的信息安全整体架构，实现涵盖防护、检测、响应和恢复等环节的全流程主动式技术防御。■

谷歌的Dart语言能否解决JavaScript的速度与规模难题？

□ 核子可乐/译

大型JavaScript Web应用程序很可能既难于开发、又运行缓慢。相比之下，谷歌的Dart语言针对这两大难题给出了解决方案。

JavaScript如今的主要应用方式与其最初定位可谓风马牛不相及：当下它正作为托管于浏览器当中的平台用于大型Web应用程序开发。如果创造JavaScript的技术人员能够早点预见到这类应用方式——而不仅仅是为Web页面增添活力——那么JavaScript的设计思路很可能完全不同。

这是因为在大型Web应用程序的开发流程中，存在着两大主要难题。

首先，性能难题：作为无法辩驳的事实，利用JavaScript编写的大型应用程序在运行速度方面相对较慢，这必然会给程序用户造成负面影响。

其次，JavaScript本身的结构也存在问题：这是一种语言，因为大型团队很难利用它组织开发工作。不同模块之间缺乏明显的结构以及强有力的联系，加之代码本身的表达意图难以付诸沟通，这给开发团队造成很大困扰。对于个人开发者，沟通便捷性似乎并不重要；但对于大型项目的开发团队而言，沟通编码意图是保证项目成功的关键所在——尤其是随着时间推移产生的人员更替，继任者很可能对遗留代码感到困惑与迷茫。

大型应用程序通常采用模块化开发机制，即由不同开发人员负责各自独立的代码片段。然而由于JavaScript的动态特性，对象行为会随着时间推移发生变化，这与C、C++、Java或者C#完全不同，意味着大家需要通过执行代码来确定其具体作用。

对Web应用运行速度的渴求

经过长时间的调整，JavaScript当然已经拥有了一些令人振奋的速度提升效果，而且五年之后横空出世的谷歌V8 JavaScript引擎为这款根基孱弱的语言带来巨大飞跃。但必须承认，JavaScript开发人员仍然受到目前这一代JavaScript引擎的严重束缚，我们恐怕还要等上很久才能迎来下一轮革命性进化。

可能的方案之一在于asm.js，这是一套采用高度限制机制的JavaScript子集，最显著的特征在于舍弃了动态特性——也就是JavaScript优化道路上的最大障碍。

顾名思义，asm.js通常被视为一种针对JavaScript引擎的汇编语言。它目前尚处于测试阶段，而且

由Mozilla负责项目推进。

由于它属于JavaScript的子集，因此拥有全面向下兼容现有JavaScript引擎的能力。另外，能够与asm.js协作的引擎在运行速度上也远高于普通JavaScript——其具体速度能够达到本地代码的一半左右。（最新版本的火狐浏览器[22版本]能够支持asm.js优化，Mozilla公司的开发人员网站上也给出了技术演示材料。）

正如C++或者C#开发人员在处理对性能要求较高的代码片段时常常选择低级语言，asm.js也可以成为JavaScript开发人员的备选方案——大部分是游戏开发者——并成为创建关键性能代码片段的理想途径。没错，不太可能会有开发人员愿意用它开发整个应用程序，但它确实能让代码中的某些子集拥有理想的运行速度。

事实上，大多数开发人员可能从来不会直接使用asm.js。他们更可能使用C或者C++（或者利用现有C或C++应用程序）而后利用Emscripten转译器将代码编译为asm.js。

大型Web应用程序开发

虽然有能力带来速度提升，但在利用JavaScript开发大型Web应用程序方面，asm.js就帮不上什么忙了。相比之下，帮得上忙的要数微软提供的TypeScript方案。从概念上讲，TyperScript与asm.js正好相反：asm.js属于JavaScript的子集，而TypeScript则属于超集。

这个超集的作用在于为JavaScript开发流程带来一定结构，从而通过命令行编译器利用插件将常规JavaScript、类型检查以及Visual Studio 2012加以集成。它的局限性在于无法带来性能提升：TypeScript编译而成的JavaScript内容与手写JavaScript内容几乎相同。

Web应用程序的速度与规模

综上所述，asm.js带来更理想的运行速度，而TypeScript则提供适用于大型Web应用程序的JavaScript开发环境。但二者无法同时解决这两大难题。

而这正是谷歌Dart语言的预定目标。根据谷歌公司内部邮件的说法，Dart希望成为一款“JavaScript的终极替代方案，正如用于Web开发的通用语在开放Web平台上的角色。”

这个目标可谓雄心勃勃，那么Dart到底是什么？它是一种开源编程语言，在设计之初就考虑到了大型应用程序开发以及高运行性能两大实际需求。事实上，将Dart称为一种语言并不贴切，因为Dart本身还捆绑有一款验证器及其它各种开发工具。由Dart编写的应用程序能够运行在Dart虚拟机当中，且运行速度可达到浏览器上JavaScript速度的两倍。目前只有谷歌的开源Chomium定制浏览器版本才支持Dart，该浏览器名为Dartium，但相信Chrome对其实现全面支持将只是时间问题。

编译为JavaScript

Dart代码可以通过dart2js编译器转化为JavaScript

内容，从而以兼容方式运行在一般浏览器当中。尽管由Dart编译而成的JavaScript代码在运行速度上无法与本地Dart代码相提并论，但它仍然比开发人员们人工编写的JavaScript代码快得多，软件开发者兼《Dart在行动》一书作者Chris Buckett解释道。

“当代码由Dart转化为JavaScript时，编译器所做的类似于摇动树干以震下枝叶，”Bucket指出。在JavaScript方面，即使只需要其中一项功能、大家也不得不添加一整套库。但在“摇树”原则的帮助下，Dart能够在向JavaScript转化的过程中只纳入必要的单项功能而非完整库。Dart还会对我们的代码进行分析并有选择地进行类型检查，从而删除一部分不必存在的内容，Buckett表示。而在人工编写的JavaScript代码中，我们将被迫以非常保守的方式处理内容。

杀手级Dart应用

目前除了谷歌公司之外，还没有哪家浏览器供应商愿意直接为Dart提供支持，因此Dart成为另一种Web通用语言的可能机微乎其微。不过Buckett认为，杰出性能与简易规模化开发两大优势——外加Dart工具的现有生态系统——很可能会推动Dart走上普及之路。

“在短期内我们还不太可能在其它浏览器中看到Dart虚拟机的出现，但如果Dart能够拿出几款杀手级应用程序成品，而且在Chrome当中飞速运行——例如下一个Facebook版本——那么人们很可能希望能在自己的浏览器中看到Dart的身影，”Buckett提出假设。“这种态势在移动浏览器领域出现的可能性更高，因为JavaScript正是移动设备电池寿命的主要威胁之一。”

现在JavaScript开发人员还迎来了另一条好消息——从JavaScript到Dart的技能转化过程非常简单，不过Dart的编程要求比JavaScript要更严格一些。总而言之，任何一位曾经利用Java或者C#等服务器端语言从事过开发工作的技术人员都能够打理好这些额外限制，因此Dart的未来可谓一片光明。■

原文链接：
http://developer.51cto.com/art/201309/409990.htm

热门资源

40个让你的网站更加友好的jQuery插件

20个势头最猛的开发者工具

35个免费创新的响应式HTML5模板

15个实用HTML5、JS工具和jQuery插件

10个免费超级有用的扁平UI工具包

20款绝佳的jQuery倒计时脚本和插件

10 款最新且超实用的 Web 开发框架

10个最佳的免费项目管理工具

推荐15个适合用于Web设计的字体

推荐！25个应用圆形元素的网页设计

WebKit最新特性srcset简介

□ 廖煜嵘/译

WebKit内核最新新增了对srcset属性的支持（参考：https://www.webkit.org/blog/2910/improved-support-for-high-resolution-displays-with-the-srcset-image-attribute/）,这是首个宣布支持srcset的浏览器引擎。

srcset属性是由W3C旗下的响应式图片社区（http://responsiveimages.org/）首先提出的，目的是旨在为使用不同分辨率的不同浏览器用户提供适合其浏览环境的图片大小的解决方案。

作为W3C响应式图片社区的主席，我本人对此功能期待已经有很长的时间了。现在该属性率先由Webkit内核宣布支持，这是个天大的好消息，而且对参与的各方　　无论是用户还是浏览器厂商来说都是很有利的。在本文中，我将简单介绍关于srcset属性。

```
<img src=" low-res.jpg"  srcset=" high-res.jpg 2x" >
```

srcset属性的目的在于允许开发者为某个图片的属性指定一系列的来源，其中这些图片的来源是要根据客户显示屏的像素分辨率而设定的，比如：

```
<img src=" low-res.jpg"  srcset=" high-res.jpg 2x" >
```

其中指出了使用图片作为在低分辨率的显示屏上显示的默认图片以及不能识别srcset属性的浏览器中也会使用该图片；而srcset中指定的图片，则会在能识别srcset属性的浏览器中同时是高分辨率屏幕中显示。可以看到，其语法类似于苹果对Retina-ready图形卡的定义：开发人员只需提供一个备用的文件名(alternate filename)和倍数放大(resolution multiplier)，比如1x、2x或4x。”Resolution Multiplier”是用来衡量”多少个物理像素组成’一个’像素点的方法”，例如iPhone 5的屏幕分辨率为1136x640，但”显示分辨率”为568x320。这意味着4个物理像素组成了一个”显示像素”，或称”4x multiplier”。

这样其中的一个好处是，持有高分辨率显示设备的用户（典型的如苹果设备的用户）能很轻松得浏览那些能提供高质量图片的网站，从而带来很好的用户体验。与此同时，那些使用普通显示分辨率的用户也不会因为网络带宽等问题而懊恼浏览高分辨率图片较多的网站，因为这些网站能提供适合他们观看的图片。

接下来，我们会有这样的疑问，我们能否使用Javascript去实现这个属性的功能？其实srcset属性所做的事情并没有太特别，它根据用户的显示分辨率从一系列的可供选择的列表中选择了某张图片，然后替换原来src属性所指定的。看上去这个步骤可以完全用Javascript去代替，但为什么要使用全新的属性去实现呢？

实际上，我们尝试对网站BostonGlobe.com进行

响应式图片设计的时候，采用过这样的方法，这个网站也是我们较早采用“响应图片”解决方案的网站之一。但由于目前几个主要的浏览器对图片具有越来越强大的预读取功能(prefetch)，因为图片被读取之前，我们很难有机会去自定义脚本，最后我们为每一个图片发起了两个不同的请求。我将其中的一些过程记录

在（http://alistapart.com/article/responsive-images-how-they-almost-worked-and-what-we-need），有兴趣的读者可以参考。

那么我们能用CSS去实现这个功能吗？我们可以使用背景图片并结合和像素有关media查询语法去实现这个功能。由WebKit实现的srcset属性和CSS 3中最新的image-set属性有点像。image-set允许指定一系列的背景图片和分辨率 并能让浏览器去判断哪一种是最适合用户的。

使用CSS去按上面的方法去管理图片在不同分辨率下的显示的话，在一些简单的比如示例性的页面上是没问题的，但如果一旦应用到生产环境的规模比较大的网站中去，是多少会出现性能上或者各种各样的问题。

从开发人员的角度看，让CMS网站去成大量的背景图片并没有特别的好处。然而，更糟糕的是，它会导致用户每一次会请求很多并不需要的额外的样式和图片（当然除非你的CSS设计的十分仔细和谨慎）。除此之外，它使得我们的图片丢失了语义，这在搜索的环境下显得不是那么有利。

最接我们已经找到了一个基于CSS的方法，能通过基于HTML5中的数据属性值的方法，更换掉图片的来源，这其中使用了一些CSS的技巧（参考http://nicolasgallagher.com/responsive-images-using-css3/），但是要注意的是，其中大部分只是理论上的，可能在生产环境上会遇到各种问题，并且，它仍然没有解决如脚本遇到的同样问题：在下载高分辨率图片的时候的多次请求问题。

接下来我们讨论带宽的问题。无论屏幕的分辨率如何，也有大量需要使用分辨率较低的图像源的情况：比如Retina的MacBook Pro的连接到3G网络环境，或不稳定的会议WiFi网络。

除了能为用户提供一种内嵌简单的分辨率媒体查询功能外，srcset属性也在一定程度上考虑了带宽。真正令人兴奋的是，srcset是它定义为一组提供给浏览器的建议方案。然后，浏览器可以根据使用环境或用户的喜好去决定，它到底是使用一个较低的分辨率的图片还是使用高分辨率的图片。

实际上，我们是很倾向根据用户显示设备不同的分辨率去发送图片的，因为这样既节省了带宽，也能加速图片的下载。如果熟悉HTML 5的读者可能会记得<picture>标签，那么srcset标签和这个picture标签有什么异同呢？

由WebKit所实现的srcset的版本是和原来建议的srcset功能相匹配的，也跟响应式图片社区一直致力的版本是相符的。我们可以认为这个srcset其实就是化身为针对分辨率的快速的媒体查询方法，一个关键的区别在于浏览器可以选择源根据用户的喜好和选择进行选择。

虽然这已经是匹配原来srcset草案的建议，但当前srcset规范还正在试图扩大语法涵盖的范围，其中有的部分跟<picture>标签的功能是有重合的，如：

```
<img src=" fallback.jpg"  srcset=" small.jpg 640w 1x, small-hd.jpg 640w 2x, large.jpg 1x, large-hd.jpg 2x" alt=" ..." >
```

象上面这种模式的语法在我们看来并不理想。我们限制一些和媒体查询语法中如max-width、像素和高深莫测的一些用法，其目的是尽可能能重用媒体查询语法的功能。幸运的是，web开发人员或者浏览器厂商都不是特别喜欢过度复杂的语法。

而<picture>标签的存在其目的是为了能用更灵活的和熟悉的语法，去解决一些问题例。<picture>标签在source元素中可以使用media属性，和video标签类似。这使我们能够针对图像源做一些组合：viewport的高度和宽度，以像素或ems为单位，使用min或max值，就和我们使用CSS media查询一样。

```
1. <picture>
2.    <source src=" med.jpg"  media=" (min-width: 40em)"  />
3.    <source src=" sm.jpg"  />
4.    <img src=" fallback.jpg"  alt=" "  />
5. </picture>
```

要注意的是，我们是可以在<picture>标签中使用srcset属性的，例子如下：

```
1. <picture>
2.    <source srcset=" med.jpg 1x, med-hd.jpg 2x"  media=" (min-width: 40em)"  />
3.    <source srcset=" sm.jpg 1x, sm-hd.jpg 2x"  />
4. <img src=" fallback.jpg"  alt=" "  />
5. </picture>
```

最后要注意的是，尽管Webkit在基于响应式图片的解决方案中率先行动了，但我们也期望其他浏览器继续跟上这个趋势，同时在http://usecases.responsiveimages.org/上，列举了在响应式图片处理方案的最新研究趋势。■

原文链接：
http://developer.51cto.com/art/201309/410720.htm

趣 文

如果妹纸是网络浏览器

如果女人是浏览器（What if Girls Were Internet Browsers），这个时尚话题我很久之前就想过，但一直没机会组建一个合适的团队。这需要大量的前期制作和准备，因为每种网络浏览器必须得有非常具体的元素、颜色和特征。

正如我在Facebook的一位好友 Juli 所评论：IE 浮华、Firefox 性感、Opera 典雅、Chrome 实用、Safari 时髦。

除了基于主色做相似视觉，我也尽力通过衣服、鞋子、饰品和道具来折射出每种浏览器的各自风格。设计概念从最初的草图到最终的产品，虽然过程相当长，但我非常高兴。

摄影师：Viktorija Pashuta、MUA’s Tokiko Inoue & Heiddis Ros

发型师： Tre Major

造型师：Chalia Young

模 特： Kaylen Dao、Brittany Ryan、Clancy McClain、Tracy Acholonu、Polina Frantsena

如何正确对待HTML5的安全问题

HTML5技术的出现使过去20多年一直驱动Web内容的技术注入了新的活力，作为新一代Web语言，HTML5凭借丰富应用、跨平台等特点被公认为未来网页技术的发展方向，并且被全球多家主流厂商寄予厚望，被捧为Adobe Flash的替代品，它能够更直接快速的在页面上有效地显示音频、图形和视频，不需要去加载任何插件。同时能够帮助开发者通过浏览器更加紧密模仿原生应用并运行，HTML5包含了很多新的功能，但是从安全的角度来看，也构成了一把双刃剑。

“它提供了一系列新的网站编程方式，但同时也给开发者和网站运营商提出了新的安全调整和隐私风险”， Neohapsis移动和云安全服务公司的高级安全顾问Aaron Rhodes认为。

新时代总是会带来新的关键词，“云计算”“大数据”就是现在互联网技术发展的代名词。Web应用的发展给用户带来的好处就是存储空间越来越多，譬如最近的网盘之争更是使得用户在存储方式则越是多样化。HTML5作为一个新的技术并不表示它安全的，网络应用开发工程师们以及其他的开发者们在学习新技术的同时也需要去思考安全问题，HTML5所购建的网页和其他语言编写的网页一样容易泄露一些敏感数据。

作为典型的安全问题就是HTML5在客户端的存储中，Cookie作为最佳的解决方案，其实存在着很多的致命问题。虽然开辟了一个新的领域，同时也暴露了新的漏洞。攻击者可以检索数据或者操作数据，然后被应用程序再次使用。

在众多的的安全问题中，HTML5中也给出了很多解决方案，比如通过提供本地存储和Indexed数据库等API接口使得web应用可以在浏览器中存储大量数据；因此安全性不仅涉及到存储的数据，同时也涉及到访问方式。

如何去看待HTML5的安全问题？小编认为，其实这些都是潜在的，每个新技术的兴起，都会存在安全的问题。不过从长远的角度来看，HTML5的优势仍然是非常明显，只有把HTML5的存储和处理数据能力作为标准，增强HTML5的普及和应用，更多的安全漏洞才会暴露出来并修正，未来才能够更加安全。当然了，未雨绸缪地制定HTML5的安全保护策略也是非常有必要的。

HTML5作为下一代的Web技术已经成为了不争的事实，完善HTML5的安全，需要坚持以及推动标准研制。不管是坚持照传统编号的版本控制系统构建一个静态快照版标准的W3C还是致力于为HTML5制定一个不断进化的动态标准的WHAT工作组，我们只要在HTML5技术上积极跟踪W3C以及WHATWG相关工作组研究，参与HTML5安全相关标准制定，就会不断的推动HTML5安全标准完善。

当然，浏览器厂商也是HTMl5标准的主要推动者，只有国内浏览器厂商参与国际标准化工作，积极推进自主创新技术的标准化和应用，HTML5才会更加安全。■

原文链接：
http://developer.51cto.com/art/201309/410665.htm

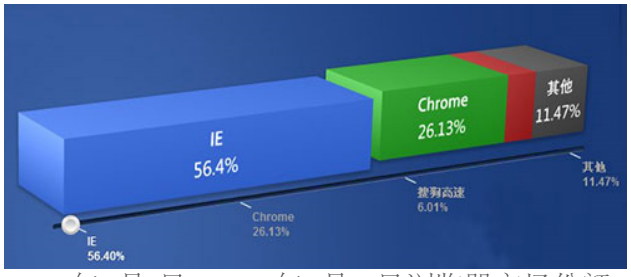
编者按

网络时代的今日，浏览器成为我们每日必不可少的工具。琳琅满目的浏览器市场，飞速的浏览器更新速度，让我们有了更多的选择和更挑剔的体验要求。

开启浏览器自主内核时代

网络时代的今日，浏览器成为我们每日必不可少的工具。琳琅满目的浏览器市场，飞速的浏览器更新速度，让我们有了更多的选择和更挑剔的体验要求。无论是国产的还是外来的，浏览器的体验要求都集中满足启动速度快、不轻易崩溃、带有常用的辅助工具；对于开发者来说，还要带有供开发者开发的相关工具。

1995年IE诞生、2004年火狐面世、2008年谷歌推Chrome，一个个“外籍”浏览器凭借优质操作体验和视觉效果，以雷霆之势占据中国浏览器市场几乎整个江山。小编获取的最近一个月浏览器市场份额显示如下图所示：



2013年8月1日—2013年8月31日浏览器市场份额

可以看出IE浏览器以绝对的明显优势占据了浏览器市场一半多份额，Chrome以26.13占据第二位，两大浏览器领跑浏览器市场；搜狗浏览器位列第三。算得上分得一杯羹；虽然前三名有国产浏览器上榜，但我们看到市场份额少的可怜；11.47%的其他成分中FireFox、360、遨游、世界之窗等混战其中。这样的局势似乎已经持续很久，国产浏览器的弊端在于基于IE、WebKit内核做浏览器，一点不接触内核部分；因此，浏览器问题的解决上依赖性过强、束手束脚。

前段时间，小编去了遨游浏览器的公司遨游天下，了解了遨游浏览器”自主内核“的开发理念、开发进度和最新进展。

为何开始自主开发内核

经历了傲游1.x、2.x以IE为内核的时代，傲游在傲游3.x选择了IE与Webkit双核为核心，开始自主开发自己的浏览器内核。

基于IE内核开发的傲游1.x、2.x，尤其是2.x版本，在用户体验上出现的问题，因为依赖于IE内核而无法自行解决。浏览器容易崩溃、错误修复后仍频繁出现、右键无法使用都是在傲游1.x 和2.x中会经常出现的问题。因此，从3.x，“自主内核”开始，将会提高浏览器的独立性、更自由来解决问题。除此之外，在短时间内，产品的创新和优化所带来的优势和利益，很容易被竞争对手copy而大大降低。可以说，对遨游而 言，自主开发内核，可谓一举两得，明智大胆的一步棋。更多的接触底层标准、变化，带来在短时间内难以复制的创新产品，带来先发的和持续的竞争优势；同时，大大提升用户体验、方便操作。

现在的傲游

现在的遨游浏览器，称之为”遨游云浏览器“

。在后PC时代，多移动设备让云计算将成为数字生活的中心，它用于承载应用、内容和参数设置，它让设备间的同步成为可能，让服务变得更为重要。最新版的傲游浏览器，最新推出的云推送功能和浏览页的二维码扫描，实现了PC端到移动端的快速转移，让我们不难看出傲游浏览器的理念：在未来提供非常强大的云服务，以及内容服务。除了云推送这个新鲜诱人的新功能，同时还有支持超级拖拽和html5拖拽、比Safari阅读模式更强大的免翻页阅读模式、服务代理器切换等强大功能。

最新进展

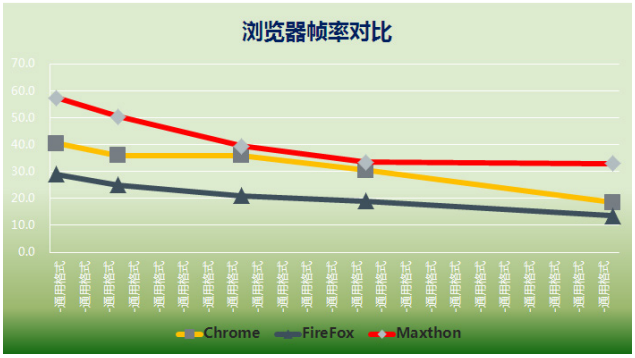
什么是WebGL

WebGL是一种3D绘图标准，这种绘图技术标准允许把JavaScript和OpenGL ES 2.0结合在一起，通过增加OpenGL ES 2.0的一个JavaScript绑定，WebGL可以为HTML5 Canvas提供硬件3D加速渲染，这样Web开发人员就可以借助系统显卡来在浏览器里更流畅地展示3D场景和模型了，还能创建复杂的导航和数据视觉化。

WebGL的作用

WebGL完美地解决了现有的Web交互式三维动画的两个问题：第一，它通过HTML脚本本身实现Web交互式三维动画的制作，无需任何浏览器插件支持;第二，它利用底层的图形硬件加速功能进行的图形渲染，是通过统一的、标准的、跨平台的OpenGL接口实现的。

目前WebGL标准已出现在Mozilla Firefox、Apple Safari及开发者预览版Google Chrome等浏览器中，遨游浏览器的最新进度中，也开始加入WebGL标准，WebGL Test的通过率高达97.5%。在和Chrome和FireFox浏览器帧频率这一属性的对比上，也呈现了明显的优势。有兴趣的网友可以进入该网站进行验证：http://aleksandarrodic.com/p/jellyfish



除此之外，遨游浏览器将在Image Viewer、第三方Cookie策略、Ogg Opus Support 等发面提供支持。现在，那个在过去被大家称作”壳“公司的遨游，也许已经渐渐远去了。未来的遨游浏览在PC端和移动端的浏览器的新创意和产品，都让人值得期待。

总结：

目前的中国浏览器市场一家独大，IE依旧势不可挡。俗话说，冰冻三尺非一日之寒，国产浏览器的”破冰“之旅将会异常艰苦。遨游浏览器的”自主内核“算是为中国浏览器的“相对独立”指明了方向，也勇敢的迈出了一大步，而这一大步的震慑力究竟如何，只能说我们都在观望之中，毕竟这项工程做的人少、成功的更少。未来是否会有更多的国产浏览器厂商加入这场”破冰“之旅，我们不得而知，但从国产浏览器的发展现状看，想要做出更好的符合中国人操作体验的浏览器，不束缚于外来浏览器内核，“自主内核”是必然的一步。■

原文链接：
http://developer.51cto.com/art/201309/409830.htm

帮助CIO在IT乱世中顽强生存的四大战略

■ 各个行业的CIO们正面临着前所未有的变化性、不确定性、复杂性以及模糊性，而带来这一切的则是日益成熟的网络攻击、消费级技术以及瞬间万变的隐私规则与行业法规。

各个行业的CIO们正面临着前所未有的变化性、不确定性、复杂性以及模糊性，而带来这一切的则是日益成熟的网络攻击、消费级技术以及瞬间万变的隐私规则与行业法规。在本文中，我们将与大家分享四种应对策略，从而帮助技术领导者在乱世中顽强生存。

Stuart kippelman十岁的女儿每一次跟爸爸一起钻进汽车，都坚持认为车里的蓝牙系统出了问题。蓝上系统每次都需要花上十秒左右来同步播放设备中的音乐，而对于一个正处于青春期的小姑娘来说，十秒钟就像十年那么长。

当然，今天的年轻人将成为明天消费群体的中坚力量。“他们对即时性要求很高，而IT人士必须为此拿出令人满意的方案，”现任Covanta能源公司CIO兼Computerworld网站博主的Kippelman指出。

但对速度的需求还只是冰山一角。在各个行业中，IT团队正面临着前所未有的变化性、不确定性、复杂性以及模糊性，这四种特性拥有一个新的缩写——VUCA。这是个源自军事领域的术语，而且恰如其分地总结了CIO在当今动荡商业环境中所需要面临的日常课题。

“VUCA的涵盖范围极广，既包括货币贬值、自然灾害，也涉及IT领域中的大数据与日益泛滥的网络攻击，”宝洁公司全球业务服务部门高级副总裁Linda Clement-Holmes表示。“而我们必须搞定这些难题。”

这样的例子不胜枚举：由于廉价与高效技术的逐步普及，新的竞争者们能够从很高的起点向老牌从业者发起冲击；全球隐私规则与行业规范正持续改变；而用户的预期——很大程度上源自他们从消费级电子技术身上获得的体验——也水涨船高。

“在旧有商业模式当中，大鱼总能吃下小鱼，”旧金山州立大学管理学教授、曾任Agilent技术公司前首席人才官的John Sullivan指出。但在VUCA的世界中，“快鱼吃慢鱼”才是真理。他解释称，“Facebook才刚刚创立六年，而如今它已经成为一家价值数十亿美元、由一位连大学文凭都没有的连帽衫小子所构建的社交巨头。原先，我们总能搞清认证才是自己的竞争对手；但时至今日，很多对手会一夜之间现身市场，并在短期之内将老牌劲旅打得头破血流。”

而在今天的文章中，我希望与大家分享VUCA时代下的IT生存指南。

1. 熟练掌握灵活性

在技术乱世之下，只有极强的适应性能帮助我们杀出一条血路，这已经成为IT领导者们的共识。抛开什么五年规划，在VUCA的世界里远景预

期根本一文不值。别再关注什么价值数百万美元的项目与技术投资，转而依靠IT服务努力打造“轻量级资产”，从而拥有达则迅猛发展、穷则随手即抛的灵活业务状态。而最重要的一点（听起来似乎有些违背常理）：应该同时着手几种相互矛盾的发展目标。

在VUCA的世界中，“过去需要花掉五年时间才能实现的目标如今可能只需六个月。因为我们无法再专注于同一条发展路线，而应该同时走上通往多个目标的前进轨道，”Sullivan指出。“这种有些精神分裂的发展方式必须成为未来的运营常态。”

CIO们认为，VUCA影响着一切事物——从我们IT组织的方式到雇佣人才、从削弱成本的手段到提升生产力的措施，甚至连新的产品与服务营收增长点也不例外。贝斯以色列女执事医疗中心与哈佛医学院CIO John Halamka也表示，VUCA的兴起为创新带来又一个黄金时代。

2. 乱世中寻找机遇

在医疗领域，平价医疗法案、隐私立法与国际疾病统计分类（简称ICD）曾经作为至高无上的圣经，用于指导几乎所有诊断及医疗流程分类工作。“但现在，我们的业务需要迎来一轮颠覆性的重新定义，”Computerworld网站专栏作家Halamka指出。

“举例来说，奥巴马提出的医疗法案会根据医疗中心的质量与康复结果分配资金投入，而不再像过去那样根据运营方式决定，”他解释称。“医院方面了解人们生病时应如何加以照料，但却不熟悉如何在生病之前予以关注。这就是VUCA的典型体现。”

VUCA之所以带来机会，是因为“行业中还没有人了解如何把这种思路变成现实，”Halamka认为。

“这是一个为创新者与冒险者所准备的巨大宝藏，”他补充称。再有，“新一代工具与技术的出现意味着我们现在已经有能力在这类业务流程中取得阶段性胜利。”

两年之前，大家还不清楚各种建议性规则及法规最终将带来怎样的影响。因此Halamka和他的团队推出了一套名为“猜测性培训”的方案，开始借此将所有数据汇总到贝斯以色列女执事医疗中心的集中式护理管理库当中。如今，这套信息库已经成为医疗中心电子健康记录系统与信息交换系统的重要基础。

现在，贝斯以色列女执事医疗中心正努力帮助医生解决超过十七万条计费代码，旨在遵循将于明年十月份正式生效的ICD修订法案。

“医生们将以完全不同的方式进行记录，因此我们提出了这样几个问题：‘一位医生能否记住十七万条代码？’‘我们如何利用自然语言处理等技术打破现有障碍，让计算机能够读取医生手写的内容并转换成代码？’”

“我们以自然方式对临床文档的处理方式进行了重新审视，并以一年为期对其加以调整，”Halamka告诉我们。

要想在逐步普及的VUCA环境中取得成功，必须首先实现两大目标，他解释称。首先是不再由于管理工作对敏捷性所提出的要求而感到沮丧，反而应当从中获得动力。第二，建立一个“极具弹性”的资深团队，而Halamka正是这样形容自己的技术班底。

“我们已经了解到，每一次新项目或者新任务

出现，大家都不会以‘不是我的问题，我也是受害者’为借口推托逃避。相反，大家会研究项目可行性，深入了解如何将其融入当前环境之中。这要求我们几乎每天都对重点进行动态调整。”

贝斯以色列女执事医疗中心快速实现自带设备方案普及就是一个典型的例子。“我们一共支持七千台iPhone与两千台iPad，但我们一台自有设备也没买过，”Halamka指出。“我们需要快速在安全性、易用性与合规保密性之间找到创新层面上的平衡点。这意味着需要再次投入资金及人力，从安全角度出发重新制定运营规划。这一切都与医疗中心最初的设计思路不同，但时代背景的推动让我们不能坐以待毙。”

VUCA时代下的IT

快速执行、打破陈规

如果Yodle始终坚持其原有商业模式，那么这家企业早在八年前就已经一败涂地。然而时至今日，它已经成长为一家拥有1.3亿美元市场份额的技术与服务公司，并以欣欣向荣的态势继续发展。

“这实际上是促使我们建立系统与流程的原动力，”公司CTO John Merryman指出。“许多高科技新兴企业都面临着这样的难题：不知道下一步该往哪走。”换句话来说，VUCA已经成为日常业务中的组成部分。

举例来说，团购的雏形就源自帮助企业尽早筹集资金的需求，并随即转化为如今的团体优惠券形式，Merryman指出。Facebook目前仍然在调整自身的收入模式。“尽可能多地做出尝试才能帮自己找到最佳答案，”他认为。“只把已经做到的事情做得更好，根本不足以扭转乾坤。”

一言以蔽之，Yodle是数字化世界为旧黄页时代开出的一剂良方。当整个世界走上网络平台，大型企业自然有能力第一时间建立并运营自己的网站。但小型地方性企业则有心无力。在Yodle公司的帮助下，本地企业终于找到了利用现有技术平台快速建立网站的途径。

但首先，Yodle推出的是一套在线企业目录，也就是名为Yodle Local的服务。

“我们的想法在于建立直接消费关系及开发领导地位。我们作出尝试，而且确实获得了一些振奋人心的回馈——但谷歌随后改变了搜索算法，这使得搜索结果倾向于减少名录之类的内容，导致我们的工作成果受到严重影响，”Merryman表示。

有鉴于此，该公司迅速调整了业务路线，开始以分散化IT形式保持业务敏捷性。

“每一家新兴企业都具备出色的灵活性，对这类灵活企业进行规模调整时，必须会出现一些必须进行变更的拐点，”他解释道。“我们取得成功的关键在于，我认为敏捷性在创建产品及服务业务部门的过程中起着至关重要的作用。”

为此，Yodle的IT部门并没有制定什么“全局优先级列表”。

“一大堆相对优先级项目会让事情变得难以权衡，例如满足合规审计要求与创建能为企业带来上百万美元收益的新产品功能之间，到底哪一边更重要？”Merryman解释称。“人们往往会把时间浪费在争论上，而无法切实开展工作。”

相反，Yodle公司的IT部门被划分为多个功能团队，每一个负责特定的业务区块。

“业务区块中的相对支出不会出现大规模浮

动，真正改变的是资金的具体去向。就以服务营销领域为例，这里随时可能出现多种多样的想法，而且由于他们拥有自己的独立优先级列表，因此能够以极低的成本快速调整工作方针，”Merryman表示。“他们对于各自领域的业务挑战非常熟悉；工程师们了解客户的需求，因为双方经常进行直接沟通。这样清晰明确的执行流程保证了方案不受误解与争议的影响。”

相比之下，“如果采用集中化调度的方式，即使大家凭借自身能力将方案推向顶峰并取得成功，来自其它方面的负面影响还是有可能让我们不得不眼睁睁看着自己的成果化为乌有，”他告诉我们。“这将导致人们不愿改变现有优先级结论。而功能团队方案保证了我们能够始终将核心资源利用在最终商业价值身上。”

总而言之，Merryman认为“我们已经通过整体迭代转换将流程送入正轨，并以合适的价格将合适的产品以合适的方式进行销售。让自己快速尝试新思路、快速了解哪些方式可行、哪些无法奏效，这就是获得成功的最大前提。”

3. 像变色龙一样生存

既然无法彻底控制环境，宝洁公司的Clement-Holmes选择了一套以能够控制的对象为基础的管理策略。

“VUCA的核心在于那些我们不了解的事物。我们希望让员工专注于那些自己确实了解而且能够控制的对象，”她指出。举例来说，与其在员工到齐之前坐等二十分钟来召开一个未来十二个月的日程安排会议，“不如让员工立即动手，相信自己处理当前难题的能力，”她建议称。

“在VUCA的世界里，大家必须一边前行一边做出决定。我们建议员工不要坐等一切条件达到百分百完美，也不要刻意把简单的问题复杂化。你的直觉和职业本能就是最好的方向指南，”她补充道。

作为VUCA世界中的一位IT领导者，Clement-Holmes认为她的图腾就是变色龙，这种蜥蜴的双眼能够分别转动并同时对两种不同的对象进行集中观察。在她看来，IT部门的作用正是在于从长期发展与短期效益之间找到平稳点，“无论是否处于VUCA环境之中，”她如是说。

“在宝洁公司，我们每年都需要为企业增加数十亿美元营收，而且全球经济衰退、欧洲经济动荡以及商品成本高企等不确定因素都给营收增长带来不确定性阻力。我们再也无法用过去的那一套理论指导如今的运营工作，”她表示。“转变的步伐也有所不同。过去我们的转变周期更长、时间也更充裕。”

与此同时，IT部门还必须积极推动成本节约并提高生产效率。这意味着我们需要缩短规划周期并不断重塑IT组织结构。

“以五年为周期预测业务走向是不现实的，”她解释道。“五年中能够发生无数变故，现实中的一年相当于IT行业的七年。”

两年前，宝洁公司创立了名为FLOW的全新体系，Clement-Holmes将这一体系形容为“一种利用专业全职员工利用相关技能解决实际业务难题的特殊机制”。

在2010年在温哥华举办的冬季奥运会上，FLOW团队的成员们快速建立起由宝洁赞助的房屋，用于帮助运动员与他们的家人团聚。“他们需要在两周时间内完成房屋构建并投付运

作，”她回忆称。“在FLOW的帮助下，我们得以在两天之内组织起拥有相关技能的优秀团队。”

她把这个临时团体比喻成医疗分类单位。“他们会评估用户的需求并将其引导至正确的地点。不过如果真的需要进行‘现场手术’这样的高难工作，他们也完全有能力搞定。”每一年都有20%的FLOW团队成员被转换到其它单位，“所以我们将拥有更多思维敏捷的杰出人才，”她总结道。

VUCA时代下的IT

CIO: 这不只是一份工作，更是一种生活方式

John Halamka表示自己绝对不会天真地认为VUCA趋势会在短时间内有所减弱。事实上，他认为这种趋势只会变得更加强烈。

“我们生活在一个全球资源不断萎缩、需求却日益强烈的时代。我们沟通的步伐正变得越来越快，”他指出。就以今年四月为例，在波士顿马拉松爆炸事件发生之后，贝斯以色列女执事医疗中心的工作人员通过Facebook、Twitter以及其它社交媒体第一时间与现场取得了联系。

因此，身为CIO，我们又该如何保持自己亲民、理智及均衡的特性？

“我每周四都会在博客上撰文讲述自己的农事活动经历，”Halamka表示，他饲养着五十头动物并照料着一大片苹果园，并把其中发生的趣事写在自己的博文当中。

“我把我的周末用在搬运木料和处理粪便等工作上，”他指出。“周六和周日每天要花十三到十四个小时处理农活，这让我能够精神抖擞地迎来周一并以清醒的头脑处理新项目、新预算以及新需求。我认为CIO这个角色并不限于一份工作，而更该被视为一种生活方式。大家必须保证这种生活方式中具备释放压力与保持心态平稳的要素。”

4. 轻资产成为主流

在总部位于伦敦的Pearson PLC公司，员工们专注于为世界各地的用户提供教育服务。全球化似乎是一个永远无法完全满足用户需求的课题，而网上学习则将VUCA发展到一个全新的层面。

“从本地市场向全球市场迈进是企业所面临的最大变化之一，”Person公司CTO Graham Calder指出。举例来说，随着全球用户对于英语培训需求的持续猛增，尤其是在巴西与中国，“我们正在以统一化方式处理全球范围内的英语教学工作。”

新型且不断增长的需求促使企业在IT战略与技术投资领域做出转变，其内容包括内部企业系统、云技术以及消费技术等多个项目。

“我们将消费技术作为我们技术战略中的核心要素，并决定迎接由云计算带来的合规性挑战，”Calder表示。这种“轻资产”计算基础设施让Pearson得以快速向新的市场发起冲击。由此带来的另一大好处在于，轻资产同时也降低了失败所引发的资源浪费，因为技术成本更加低廉、企业完全可以果断将无法正常起效的环节加以舍弃。

相比较而言，一旦大型项目被撤销、伴随而来的必然是资金投入的巨大浪费。“这使得人们难以彻底放弃已经难以挽救的错误决策，“Calder解释称。而在云计算方面，他认为“它使‘快速失败’概念成为现实，因为云属于典型的‘轻资产’方案。我们可以迅速判断这类项目是否能够切实起效，如果答案是肯定的，再行着手实

施。”

举例来说，在创建新的信息协作平台时，Calder的团队就对数项中游流程做出了变更。

“我们也许失去了两到三个星期的宝贵时间，而且没能制定出任何有意义的实质性方案。这类状况往往会定期发生，而且企业领导层可能根本毫不知情，”他指出。“在某些方面，这就是衡量成功的另一种可行方式。如果能把大型决策拆分为相对较小且更加宽松的失败评估机制，那么员工完全可以在无需经过高层行政授权的前提下拿出令人满意的执行方针。”

不过轻资产战略本身也是一种重要的思维方式转变，尤其是对行政领导们而言。为了实现成功与敏捷特性，高层管理人员必须停止将技术视为管理成本要素的习惯，转而将其看作可资利用的机遇。根据Calder的观点，“当我们开始从各个角度审视技术，就能够制定出更切合实际的治理策略。作为IT领导者，大家必须信任自己的员工以及他们根据知识储备所做出的选择。”

“CIO们过去一直在谈论企业级技术。现在，一切都要以消费级为标杆，”位于弗吉尼亚州夏洛茨维尔的通用电气智能平台部门CIO兼精益主管Vince Campisi表示。“如今的重点在于，技术能否在消费环境之下经受住考验。Twitter与Facebook已经成为人们进行沟通的主要方式，而它们的出现给行业的运作方式带来了实实在在的改变，”他提醒道。“现在，我们需要拿出企业领导者所具备的工作积极性与理解能力，关注这些新生事物将给行业带来怎样的影响。”

在Covanta公司，快速失败与项目交付已经在如今的VUCA环境中成为实现商业价值的IT战略基石，Kippelman告诉我们。

在过去的两年里，IT部门利用一套名为QlikView的工具创建并交付了接近三十款应用程序。每款应用程序的开发周期都不超过两星期。

“我宁愿花七到八个月发布十款产品，也不愿意把这么长的一段时间都花在同一款产品身上，”他解释称。“在这样一个Facebook每天对软件进行数十次更新的时代下，我认为企业必须学会理解并接受这种新趋势并为其提供更多支持。”■

原文链接：
http://developer.51cto.com/art/201309/410189.htm

链 接

Java 8发出提示 可能将不再支持Windows XP

今天安装Java JDK 8 108版时安装程序弹出提示，大意是 Java 8 需要更新版本的Windows。你可以继续安装但是为了Java能正常的运行，我们建议你升级你电脑的操作系统。

值得注意的是，上一个JDK8的预览版106版还没有提示，今天发布的108版开始提示要去升级系统，这意味着等明年年初Java8正式发布时很可能不再支持XP系统。■

Java新手入门的30个基本概念

aijuans /文

在我们学习Java的过程中,掌握其中的基本概念对我们的学习无论是J2SE,J2EE,J2ME都是很重要的,J2SE是Java的基础,所以有必要对其中的基本概念做以归纳,以便大家在以后的学习过程中更好的理解java的精髓,在此我总结了30条基本的概念。

Java概述:

目前Java主要应用于中间件的开发(middleware)---处理客户机于服务器之间的通信技术,早期的实践证明,Java不适合pc应用程序的开发,其发展逐渐变成在开发手持设备,互联网信息站,及车载计算机的开发.Java于其他语言所不同的是程序运行时提供了平台的独立性,称许可以在windows,solaris,linux其他操作系统上使用完全相同的代码.Java的语法与C++语法类似,C++/C程序员很容易掌握,而且Java是完全的彻底的面向对象的,其中提出了很好的GC(Garbage Collector)垃圾处理机制,防止内存溢出。

Java的白皮书为我们提出了Java语言的11个关键特性。

(1)Easy:Java的语法比C++的相对简单,另一个方面就是Java能使软件在很小的机器上运行,基础解释其和类库的支持的大小约为40kb,增加基本的标准库和线程支持的内存需要增加125kb。

(2)分布式:Java带有很强大的TCP/IP协议族的例程库,Java应用程序能够通过URL来穿过网络来访问远程对象,由于servlet机制的出现,使Java编程非常的高效,现在许多的大的web server都支持servlet。

(3)OO:面向对象设计是把重点放在对象及对象的接口上的一个编程技术.其面向对象和C++有很多不同,在与多重继承的处理及Java的原类模型。

(4)健壮特性:Java采取了一个安全指针模型,能减小重写内存和数据崩溃的可能型。

(5)安全:Java用来设计网路和分布系统,这带来了新的安全问题,Java可以用来构建防病毒和防攻击的System.事实证明Java在防毒这一方面做的比较好。

(6)中立体系结构:Java编译其生成体系结构中立的目标文件格式可以在很多处理器上执行,编译器产生的指令字节码(Javabytecode)实现此特性,此字节码可以在任何机器上解释执行。

(7)可移植性:Java中对基本数据结构类型的大小和算法都有严格的规定所以可移植性很好。

(8)多线程:Java处理多线程的过程很简单,Java把多线程实现交给底下操作系统或线程程序完成.所以多线程是Java作为服务器端开发语言的流行原因之一。

(9)Applet和servlet:能够在网页上执行的程序叫Applet,需要支持Java的浏览器很多,而applet支持动态的网页,这是很多其他语言所不能做到的。

基本概念:

1.OOP中唯一关系的是对象的接口是什么,就像计算机的销售商她不管电源内部结构是怎样的,他只关系能否给你提供电就行了,也就是只要知道can or not而不是how and why.所有的程序是由一定的属性和行为对象组成的,不同的对象的访问通过函数调用来完成,对象间所有的交流都是通过方法调用,通过对封装对象数据,很大限度上提高复用率。

2.OOP中最重要的思想是类,类是模板是蓝图,从类中构造一个对象,即创建了这个类的一个实例(instance)。

3.封装:就是把数据和行为结合起在一个包中)并对对象使用者隐藏数据的实现过程,一个对象中的数据叫他的实例字段(instance field)。

4.通过扩展一个类来获得一个新类叫继承(inheritance),而所有的类都是由Object根超类扩展而得,根超类下文会做介绍。

5.对象的3个主要特性

◆ behavior---说明这个对象能做什么.

◆ state---当对象施加方法时对象的反映.

◆ identity---与其他相似行为对象的区分标志.每个对象有唯一的indentity 而这3者之间相互影响.

6.类之间的关系:

◆ use-a :依赖关系

◆ has-a :聚合关系

◆ is-a :继承关系--例:A类继承了B类,此时A类不仅有了B类的方法,还有其自己的方法.(个性存在于共性中)

7.构造对象使用构造器:构造器的提出,构造器是一种特殊的方法,构造对象并对其初始化。

例:Data类的构造器叫Data

new Data()---构造一个新对象,且初始化当前时间.

Data happyday=new Data()---把一个对象赋值给一个变量happyday,从而使该对象能够多次使用,此处要声明的使变量与对象变量二者是不同的.new返回的值是一个引用。

构造器特点:构造器可以有0个,一个或多个参数

构造器和类有相同的名字

一个类可以有多个构造器

构造器没有返回值

构造器总是和new运算符一起使用.

8.重载:当多个方法具有相同的名字而含有不同的参数时,便发生重载.编译器必须挑选出调用哪个方法。

9.包(package)Java允许把一个或多个类收集在一起成为一组,称作包,以便于组织任务,标准Java库分为许多包.java.lang java.util java,net等,包是分层次的所有的java包都在java和javax包层次内。

10.继承思想:允许在已经存在的类的基础上构建新的类,当你继承一个已经存在的类时,那么你就复用了这个类的方法和字段,同时你可以在新类中添加新的方法和字段。

11.扩展类:扩展类充分体现了is-a的继承关系.形式为:class (子类) extends (基类)。

12.多态:在java中,对象变量是多态的.而java中不

支持多重继承。

13.动态绑定:调用对象方法的机制。

编译器检查对象声明的类型和方法名。

编译器检查方法调用的参数类型。

静态绑定:若方法类型为priavte static final 编译器会准确知道该调用哪个方法。

当程序运行并且使用动态绑定来调用一个方法时,那么虚拟机必须调用x所指向的对象的实际类型相匹配的方法版本。

动态绑定:是很重要的特性,它能使程序变得可扩展而不需要重编译已存代码。

14.final类:为防止他人从你的类上派生新类,此类是不可扩展的。

15.动态调用比静态调用花费的时间要长。

16.抽象类:规定一个或多个抽象方法的类本身必须定义为abstract。

例: public abstract string getDescripition■

本文链接：
http://developer.51cto.com/art/201309/409236.htm

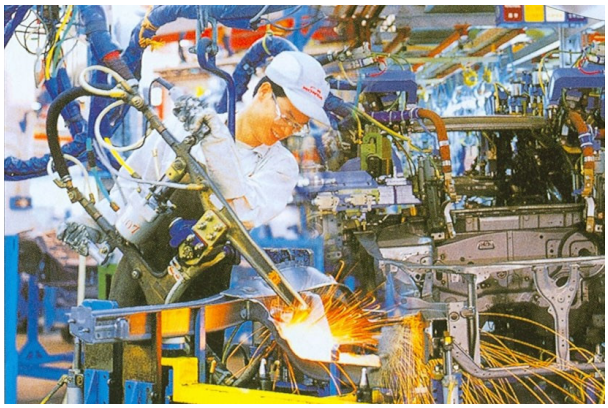
期待已久：Java终于支持白名单

甲骨文为Java添加一个让大家久等的功能“Deployment Rule Set”（部署规则集），即支持白名单。Java 7 Update 40允许系统管理员定义哪些Java程序是值得信任的，更便于管理Java安全。

很多个人用户为了防止受到针对Java攻击的影响，而选择在浏览器中禁用Java插件，甚至是卸载Java。但是，这对于大多数企业用户来说是不可行的。

因为兼容性问题，很多企业无法及时升级到最新的Java版本，这更是增加了受攻击的危险。安全研究人员批评甲骨文没有为Java提供白名单功能，这样管理员就可以规定终端用户在浏览器内仅运行特定的Java程序。

这次新增的“Deployment Rule Set”可以让系统管理员创建一个XML文件，添加信任的Java RIAs（富互联网应用）。■



一篇鞭策程序员的短文：我们这一代的汽车工人

译者按：Greg Baugues 把今日的程序员和 20 世纪 60 年代的底特律汽车工人作比较，阐述了在景气的行业下暗藏的威胁：开发者的需求越来越少，并且新开发者越来越多的现状。虽然国内外的环境有些不同，但也不难理解并深有同感的样子。

50年前，一位汽车工人足以养活一个中产阶级的家庭。买一间房子，让孩子上大学，老婆做家庭主妇，他自己甚至不用拿到学位。

这特么的已经结束了，底特律已经破产了。

现在没有任何人能够过得比开发者更好了。当你最经常听见的抱怨是“我希望招聘官别再给我发六位数offer的垃圾邮件了”之时，那生活已经变得很好了。

在历这个特殊历史时刻，开发者供不应求。没有足够的人员来流转，于是公司为了人才而掐架：超高的薪水、漂亮的办公室、自由的工作时间、内部厨师。

打破劳动力市场，要么增加供给，要么减少需求。

当汽车业引入自动化的时候，后者发生了。现在，你可以使用Shopify（一个电子商务平台——译者注）每月花$30来代替十年前需要花$500,000的定制开发。使用WordPress在十五分钟内可以完成曾经需要一名自由职业者忙两个月才能完成的工作。Stripe（一个在线付款处理系统——译者注）把整合信用卡的费用降低了5位数。

在人才供给方面。Dev Bootcamp 每9个星期便创造出十几个初级开发的应聘者。其他还有Starter League、 gSchool、 Hacker School。想自学？还有Treehouse、Code School、 Codecademy。

或许你不认为一个刚刚培训九星期的菜鸟可以做你的工作。但是一个4年花了80000美元拿到的计算机科学学位就可以去工作了。这两者之间，总有点什么。进入我们这行的门槛正在降低，甚至保护我们地位如此之久的程序员的社会烙印也会慢慢不复存在。

人无千日好，花无百日红，没有哪一行会永远风光下去……只要去问问最近毕业的律师朋友就知道了。（译注：律师躺枪）

别太安逸了，别只把自己的眼光局限在一种语言，别为了短期利益断了自己的后路。秣兵历马（指别荒废开发技能——译注），学习软技能（指情商、人际关系处理能力、沟通能力等——译注），建立读者群，存点钱，建立人际网络，多多阅读。

对于一个开发者来说，这是一个大好时机。趁着还在，多多享受吧。■

本文链接：
http://developer.51cto.com/art/201309/409782.htm

站着编程两年后我身体上的变化

■ 自从我使用站立式电脑桌工作以来已经有2年时间。不论一天要编程多少个小时，我都是站立在电脑前。也就是说，有些日子我会一天站立超过10个小时……

自从我使用站立式电脑桌工作以来已经有2年时间。不论一天要编程多少个小时，我都是站立在电脑前。也就是说，有些日子我会一天站立超过10个小时，虽然不是连续的——中间会有小憩，吃饭，冥想等。

我在决定站着编程前并没有侧过血质或其它身体指标，如今也没有测过，但下面是我自己对身体上变化的感觉。我不能把这些所有变化都归功于站着工作的功劳，每天在工作之余我都会锻炼身体。然而，站着的时间远超其它活动的时间，所以，它对我身体的改造应该比其它运动更明显。

站立工作后一些我担心会有但实际上并未出现的事情。

◆ 我的膝盖、脚、背、臀部并没有发生任何病痛症。

◆ 在一天结束时或周末时间我并没有感到精疲力尽。

◆ 我的工作效率和注意力并没有降低。

站立式工作后真正发生的变化

◆ 我的身姿比以前更好了。我的脖子和肩头不再向前曲。

◆ 我腿上有了更多的肌肉。

◆ 我不再有腰痛背痛。

◆ 我工作期间身体有了活跃的运动。

站着编程的副作用

◆ 负面效果：连续坐两个小时我就会觉得有点不舒服。

◆ 积极作用：做地铁排队时我很少再有打不起精神的感觉。我猜想以前的这种感觉部分是因为站着容易疲劳。现在不再有这种感觉了。

这两年来我的自我调整：

◆ 以前我会站在一个厚地垫上，并穿着有软鞋底的鞋。现在，不用软鞋了，也不需要地垫了。

◆ 相比起最初，我现在把笔记本升高了5英寸，因为之前的高度我的头喜欢往前下方伸。现在键盘高度在我的胸部左右，我的眼睛视角是大概前下方105度。

◆ 之前我会坐下来休息，而现在我做跳跃动作替代休息。跳一跳会让我的腿更有活力，比坐下来的效果更好。

总之，这些变化让我感到惊奇和高兴。站了两年之后，我仍然衷心的向大家推荐使用站立式电脑桌。尤其是站立式笔记本工作桌。你从这里可以找到一个。■

本文链接：
http://developer.51cto.com/art/201308/408938.htm

编者按

这是10个最佳实践的列表，比你平时在Josh Bloch的《effective java》中看到的规则更加精妙。和Josh Bloch列出的非常容易学习的、和日常情况息息相关的实践相比，这个列表中提到了一些关于……

10个Java编码中微妙的最佳实践

这是10个最佳实践的列表，比你平时在Josh Bloch的《effective java》中看到的规则更加精妙。和Josh Bloch列出的非常容易学习的、和日常情况息息相关的实践相比，这个列表中提到了一些关于设计API/SPI的实践，虽然不常见，但是存在很大的效率问题。

我在编写和维护jOOQ（一种内部DSL，在java中将SQL模块化）时，碰到了这些问题。作为内部DSL，jOOQ最大限度的挑战了java编译器和泛型，把泛型，变量和重载结合到了一起。这种太宽泛的API是Josh Bloch相当不推荐的。

让我来和你分享这10个java编码中微妙的最佳实践:

1.牢记C++的析构函数

还记得C++中的析构函数吗？不记得了？或许你真的很幸运，因为你再也不必为删除对象后，没有及时释放内存而造成内存泄露进行调试了。我们真的应该感谢Sun和Oracle实现垃圾回收机制。

尽管如此，对于我们来说，析构函数仍然有一个很有趣的特点。它常常会让我们对以和分配内存相反的顺序释放内存的工作模式感到容易理解。同样，在JAVA代码中，当你处理如下类析构函数语法的时候，也要把这个特性牢记在心：

当使用@Before和@After但与注解时

当分配和释放JDBC资源时

当调用父类的方法时

也有其他不同的使用案例。这有一个显示如何实现事件监听的实例：

```
1. @Override
2. public void beforeEvent(EventContext e) {
3. super.beforeEvent(e);
4. // Super code before my code
5. }
6.
7. @Override
8. public void afterEvent(EventContext e) {
9. // Super code after my code
10.     super.afterEvent(e);
11.     }
```

另外一个哲学家用餐的问题，显示了这有多么的重要。

关于哲学家用餐的问题，请查看链接：http://adit.io/posts/2013-05-11-The-Dining-Philosophers-Problem-With-Ron-Swanson.html

法则：无论何时，当你使用before/after, allocate/

free, take/return语法实现逻辑时，仔细想想是否需要反序的使用after/free/return操作。

2. 不要相信你早期的SPI演进判断

为使用者提供SPI可以很容易让他们注入自定义行为到你的库/代码。当心你的SPI演进判断可能会迷惑你，让你认为(不)需要附加的参数。当然，不应该过早的添加功能。但是一旦你发布了SPI，一旦你决定遵循语义版本，当你发现你可能在某些情况下需要另外一个参数时，你将真的后悔为SPI添加了一个愚蠢的单参数方法：

```
1. interface EventListener {
2. // Bad
3. void message(String message);
4. }
```

如果你也需要消息ID和消息源，怎么办？对于上面的类型，API演进将会阻碍你添加参数。当然，有了Java8，你可以添加一个defender方法，“防御”你早期糟糕的设计决策：

```
1. interface EventListener {
2.   // Bad
3.   default void message(String message) {
4.     message(message, null, null);
5.   }
6.   // Better?
7.   void message(
8.     String message,
9.     Integer id,
10.          MessageSource source
11.        );
12.      }
```

注意很不幸defender方法不能为final。

但是比起用数十个方法污染你的SPI，使用一个上下文对象（或参数对象）好很多。

```
1. interface MessageContext {
2.   String message();
3.   Integer id();
4.   MessageSource source();
5. }
6.
7. interface EventListener {
8.   // Awesome!
9.   void message(MessageContext context);
10.      }
```

比起EventListner SPI你可以更容易演进MessageContext API，因为很少用户会实现它。

规则: 无论何时你指定SPI的时候， 考虑使用上下文/参数对象，而不要编写固定参数数量的方法。

备注: 使用特定的MessageResult类型传递结果也是一个好的想法，该类型可以通过构造器API构建。这将会为你的SPI提供更多的SPI演进灵活性。

3.避免使用匿名，局部或内部类

Swing程序员通常只要按几下快捷键即可生成成百上千的匿名类。在多数情况下，只要遵循接口、不违法SPI子类型的生命周期(SPI subtype lifecycle)，这样做也无妨。

但是不要因为一个简单的原因——它们会保存对外部类的引用，就频繁的使用匿名、局部或

者内部类。因为无论它们走到哪，外部类就得跟到哪。例如，在局部类的域外操作不当的话，那么整个对象图就会发生微妙的变化从而可能引起内存泄露。

规则：在编写匿名、局部或内部类前请三思能否将它转化为静态的或普通的顶级类，从而避免方法将它们的对象返回到更外层的域中。

注意：使用双层花括号来初始化简单对象：

```
1. new HashMap<String, String>() {{
2.   put(“1”, “a”);
3.   put(“2”, “b”);
4. }}
```

这个方法利用了JLS §8.6规范里描述的实例初始化方法(initializer)。表面上看起来不错，但实际上不提倡这种做法。因为要是使用完全独立的HashMap对象，那么实例就不会一直保存着外部对象的引用。此外，这也会让类加载器管理更多的类。

4. 现在就开始编写SAM！

Java8的脚步近了。伴随着Java8带来了lambda表达式，无论你是否喜欢。尽管你的API使用者可能会喜欢，但是你最好确保他们可以尽可能经常的使用。因此除非你的API接收简单的“标量”类型，比如int、long、String 、Date，否则让你的API尽可能经常的接收SAM。

什么是SAM？SAM是单一抽象方法[类型]。也称为函数接口，很快被注释为@FunctionalInterface。这与规则2很配，EventListener实际上就是一个SAM。最好的SAM只有一个参数，因为这将会进一步简化lambda表达式的编写。设想编写

```
1. listeners.add(c -> System.out.println(c.
   message()));
```

替代

```
2. listeners.add(new EventListener() {
3.   @Override
4.   public void message(MessageContext c) {
5.     System.out.println(c.message()));
6.   }
7. });
```

设想以SAM的方式用jOOX处理XML：

```
1. $(document)
2.   // Find elements with an ID
3.   .find(c -> $(c).id() != null)
4.   // Find their child elements
5.   .children(c -> $(c).tag().
   equals(“order”))
6.   // Print all matches
7.   .each(c -> System.out.println($(c)))
```

规则：对你的API使用者好一点儿，从现在开始编写SAM/函数接口。

备注：有许多关于Java8 lambda表达式和改善的Collections API的有趣的博客：

- http://blog.informatech.cr/2013/04/10/java-optional-objects/
- http://blog.informatech.cr/2013/03/25/java-streams-api-preview/
- http://blog.informatech.cr/2013/03/24/java-streams-preview-vs-net-linq/
- http://blog.informatech.cr/2013/03/11/java-infinite-streams/

5.避免让方法返回null

我曾写过1、2篇关于java NULLs的文章，也讲解过Java8中引入新的Optional类。从学术或实用的角度来看，这些话题还是比较有趣的。

尽管现阶段Null和NullPointerException依然是Java的硬伤，但是你仍可以设计出不会出现任何问题的API。在设计API时，应当尽可能的避免让方法返回null，因为你的用户可能会链式调用方法：

```
initialise(someArgument).calculate(data).dispatch();
```

从上面代码中可看出，任何一个方法都不应返回null。实际上，在通常情况下使用null会被认为相当的异类。像 jQuery 或 jOOX这样的库在可迭代的对象上已完全的摒弃了null。

Null通常用在延迟初始化中。在许多情况下，在不严重影响性能的条件下，延迟初始化也应该被避免。实际上，如果涉及的数据结构过于庞大，那么就要慎用延迟初始化。

规则：无论何时方法都应避免返回null。null仅用来表示“未初始化”或“不存在”的语义。

6.设计API时永远不要返回空(null)数组或List

尽管在一些情况下方法返回值为null是可以的，但是绝不要返回空数组或空集合！请看 java.io.File.list()方法，它是这样设计的：

此方法会返回一个指定目录下所有文件或目录的字符串数组。如果目录为空(empty)那么返回的数组也为空(empty)。如果指定的路径不存在或发生I/O错误，则返回null。

因此，这个方法通常要这样使用：

```
1. File directory = // ...
2. if (directory.isDirectory()) {
3.   String[] list = directory.list();
4.   if (list != null) {
5.     for (String file : list) {
6.       // ...
7.     }
8.   }
9. }
```

大家觉得null检查有必要吗？大多数I/O操作会产生IOExceptions，但这个方法却只返回了null。Null是无法存放I/O错误信息的。因此这样的设计，有以下3方面的不足： Null无助于发现错误

◆ Null无法表明I/O错误是由File实例所对应的路径不正确引起的

◆ 每个人都可能会忘记判断null情况

◆ 以集合的思维来看待问题的话，那么空的(empty)的数组或集合就是对“不存在”的最佳实现。返回空(null)数组或集合几乎是无任何实际意义的，除非用于延迟初始化。

规则：返回的数组或集合不应为null。

7. 避免状态，使用函数

HTTP的好处是无状态。所有相关的状态在每次请求和响应中转移。这是REST命名的本质：表征状态转移。在Java中这样做也很赞。当方法接收状态参数对象的时候从规则2的角度想想这件事。如果状态通过这种对象转移，而不是从外边操作状态，那么事情将会更简单。以JDBC为例。下述例子从一个存储的程序中读取一个光

标。

```
1. CallableStatement s =
2.   connection.prepareCall( “{ ? = ... }” );
3.
4. // Verbose manipulation of statement
   state:
5. s.registerOutParameter(1, cursor);
6. s.setString(2, “abc” );
7. s.execute();
8. ResultSet rs = s.getObject(1);
9. // Verbose manipulation of result set
   state:
10.      rs.next();
11.      rs.next();
```

这使得JDBC API如此的古怪。每个对象都是有状态的，难以操作。具体的说，有两个主要的问题：

◆ 在多线程环境很难正确的处理有状态的API

◆ 很难使有状态的资源全局可用，因为状态没有被描述

戏剧海报《阿甘正传》，版权1994年由派拉蒙影业公司。保留所有权利。相信上述惯例满足所谓的合理使用。

规则：更多的以函数风格实现。通过方法参数转移状态。极少操作对象状态。

8. 短路式 equals()

这是一个比较容易操作的方法。在比较复杂的对象系统中，你可以获得显著的性能提升，只要你在所有对象的equals()方法中首先进行相等判断：

```
1. @Override
2. public boolean equals(Object other) {
3.   if (this == other) return true;
4.   // 其它相等判断逻辑...
5. }
```

注意，其它短路式检查可能涉及到null值检查，所以也应当加进去:

```
1. @Override
2. public boolean equals(Object other) {
3.   if (this == other) return true;
4.   if (other == null) return false;
5.   // Rest of equality logic...
6. }
```

规则: 在你所有的equals()方法中使用短路来提升性能。

9. 尽量使方法默认为final

有些人可能不同意这一条，因为使方法默认为final与Java开发者的习惯相违背。但是如果你对代码有完全的掌控，那么使方法默认为final是肯

定没错的：

◆ 如果你确实需要覆盖（override）一个方法（你真的需要？），你仍然可以移除final关键字

◆ 你将永远不会意外地覆盖（override）任何方法

这特别适用于静态方法，在这种情况下“覆盖”（实际上是遮蔽）几乎不起作用。我最近在Apache Tika中遇到了一个很糟糕的遮蔽静态方法的例子。考虑：

◆ TaggedInputStream.get(InputStream)

◆ TikaInputStream.get(InputStream)

TikaInputStream扩展了TaggedInputStream，以一种相对不同的实现遮蔽了它的静态get()方法。

与常规方法不同，静态方法不能互相覆盖，因为调用的地方在编译时就绑定了静态方法调用。如果你不走运，你可能会意外获得错误的方法。

规则：如果你完全掌控你的API，那么使尽可能多的方法默认为final。

10. 避免方法(T…)签名

在特殊场合下使用“accept-all”变量参数方法接收一个Object...参数就没有错的：

```
void acceptAll(Object... all);
```

编写这样的方法为Java生态系统带来一点儿JavaScript的感觉。当然你可能想要根据真实的情形限制实际的类型，比如String...。因为你不想要限制太多，你可能会认为用泛型T取代Object是一个好想法：

```
void acceptAll(T... all);
```

但是不是。T总是会被推断为Object。实际上你可能仅仅认为上述方法中不能使用泛型。更重要的是你可能认为你可以重载上述方法，但是你不能：

```
1. void acceptAll(T... all);
2. void acceptAll(String message, T... all);
```

这看起来好像你可以可选地传递一个String消息到方法。但是这个调用会发生什么呢？

```
acceptAll(“Message”, 123, “abc”);
```

编译器将T推断为<? extends Serializable & Comparable<?>>，这将会使调用不明确！

所以无论何时你有一个“accept-all”签名（即使是泛型），你将永远不能类型安全地重载它。API使用者可能仅仅在走运的时候才会让编译器“偶然地”选择“正确的”限定最多的方法。但是也可能使用accept-all方法或者无法调用任何方法。

规则：如果可能，避免“accept-all”签名。如果不能，不要重载这样的方法。

结论

Java是一个野兽。不像其它更理想主义的语言，它慢慢地演进为今天的样子。这可能是一件好事，因为以Java的开发速度就已经有成百上千个警告，而且这些警告只能通过多年的经验去把握。

敬请期待更多关于这个主题的前十名列表！

■

本文链接：
http://developer.51cto.com/art/201308/408528.htm

可在广域网部署运行的QQ高仿版：GG叽叽V1.8（源码）

□ GG叽叽 /文

距离的GG 1.0发布已经三周了，这三周内，我利用业余时间为GG增加了视频聊天的功能。个人觉得进展有些缓慢，主要是因为大多数时间都花在了UI上。由于本人不会PS，所以图片素材都是从网上一个一个搜下来的，这个过程确实很烦人，而且最终有些素材还不是很满意。

一.GG V1.8 新增功能展现

（1）发送视频会话请求，对方可以接受或拒绝对话。

（2）在视频会话的过程中，可以控制自己麦克风的输出、摄像头的输出、以及扬声器是否播放对方的声音。

（3）在视频会话的过程中，可以关闭/显示自己的小窗口。

（4）在视频会话的过程中，任意一方掉线，将结束视频会话。

废话不多说，还是先上图。邀请对方进行视频会话（图一）：

被邀方（图二）：

视频会话界面（图三）：

（图三中左下侧的第二和第三个按钮，用于控制麦克风输出和扬声器的播放的，每个按钮有两种状态，所以共4个图标素材，个人觉得都不太好，希望能替换掉，如果能提供的朋友，请发到我邮箱，我会在下个版本中将其更新。）

二.实现思路

虽然提供了源代码，但是，我还是想将主要的思路列一下（包括上一版本主要功能的实现思路，上篇文章漏掉了，这里一起补上），这样，大家理解起源码来，会节省更多的时间。

（1）GG早期版本，都将假设所有在线的用户都是好友。后面的高级版本将会提供好友管理的功能。

（2）用户登录帐号和QQ一样，必须为数字。而且，GG内部是根据用户帐号的数字来自动设定其昵称和头像的。

（3）GG服务端中集成了ESFramework通信服务器和OMCS语音视频服务器，在GG客户端的配置文件中可以配置服务器的IP和端口。

（4）客户端还未实现通过UI来进行麦克风和摄像头的测试功能（后续高级版本将会提供），麦克风、摄像头以及扬声器的选择可在配置文件中指定。

三.GG V1.8 源码下载

GG V1.8 源码

注意：如果要将GG部署到广域网，则可以在服务端的配置文件中设置监听的端口（Port以及OmcsPort）；而在客户端的配置文件中，则可以指定服务器的ServerIP、ServerPort以及OmcsServerIP、OmcsServerPort。

我会努力争取2~3个星期发布一个新版本，使GG慢慢成熟起来。

大家有什么问题和建议，可以留言，也可以发送email到我邮箱：ggim2013@163.com。

如果你觉得还不错，请粉我，顺便再顶一下啊，呵呵 。■

本文链接：
http://developer.51cto.com/art/201309/409282.htm

微软Visual Studio 2013 RC版泄露

今年6月份，Visual Studio 2013预览版与Windows 8.1预览版一同发布，现在Windows 8.1RTM所有版本均已泄露，而今天，Visual Studio 2013的候选发布版本(Release Candidate)也被泄露到了网上。

Visual Studio 2013预览版为我们带来了更好的应用程序生命周期管理功能、支持内建Git、基于云端的调试、单元测试失败的信息显示和以及新的代码编辑器等众多新功能。

这次Visual Studio 2013 RC版的泄露者正是Windows 8.1 RTM的泄露者--俄罗斯的Wzor，其版本号为12.0.20824.1，抱歉的是Wzor并没有提供该版本的下载链接。

目前我们还尚不清楚Visual Studio 2013最终版本将何时发布，微软官方只表示过它将于“今年晚些时候”推出，不过有传闻称它有可能在10月18日与Windows 8.1RTM同时发布。

日本人为什么不创业？

□ 黄龙中 /爱范儿

上周在日本的时候，随一位友人去拜访他在筑波大学留学时的老师。在行前，为了遵守日本“守时”的规矩，他只告诉我这位老师是那种“一小时挣几万”的厉害角色，我们不能迟到。我不太喜欢这个开场。

聊完公务之后，中午一起用餐，才慢慢了解到，这位叫藤井义彦、70 多岁的前辈是一位传奇人物。他毕业于著名的日本私立大学——应庆大学（Keio University），期间作为交换生在斯坦福大学读书一年。斯坦福读完书后，在美国、欧洲、亚洲做背包客流浪了 7 个月，期间在中亚一个国家时，所借宿的家庭还希望把女儿嫁给他。目前他已经去过 65 个国家。

藤井义彦（Fujii Yoshihiko）跟中国也有颇深渊源，年轻的时候沿着丝绸之路一直走到吐鲁番，去过中国多个城市。2007 年成为西北工业大学客座教授。藤井在中国出版过两本书，《挑战！哈佛 AMP 留学》和《猎头——跳槽风云录》，前者讲述他 1990 年（48 岁）前往哈佛商学院取得 AMP（高级商务管理）学位的经历，后者描述他在日本著名化妆品公司佳丽宝（Kanebo）工作 30 年后对日本式企业管理哲学的思考。

藤井从中国回到日本后，担任筑波大学客座教授，2010 年辞去该职，专心打理他 1998 年创立的顾问公司 GRI（Global Leadership Coaching），为企业高层进行培训。

以上信息是我在午餐中采访他关于“日本人为什么不创业”的问题后，发现他的开阔视野和思考问题的高度，才进一步了解到的。当初没有想到，随友人来拜访的，竟是如此一位神奇人物。

关于日本人为什么不创业的问题，藤井义彦作了两方面的回答。

一是日本学生喜欢去大公司工作，而且进入大公司后，日本人对组织没有抵抗感，比较顺从公司意志，日本人比较信奉“灭私奉公”精神。中国人却不一样，中国学生想挣很多钱，都想当老板，要争第一。他认为中国的民族性格当中，也更有野心，更有闯劲，更愿意往外看。

藤井做过一些随机调查，发现 90% 的中国学生想创业，日本学生则完全相反，90% 的学生想进大企业。比如我这位友人，留学毕业后，在日本京瓷工作两年，就回国创办了一个语言学校，面向日语人群教授中文。我后来了解到，他创业的想法其实在留学生涯开始之前就已经想好。

二是日本缺乏小企业、创新企业成长的土壤。在日本，一个小企业或创业公司要成长起来非常困难。因为与欧美市场不一样，在日本发布一个创新产品，日本人会关心创业者的背景、公司的背景，看重公司的信誉度。要知道，一个创业公司从无到有，一开始几乎是没有任何名气和信誉度的，但日本社会却很在意这些。

由于缺乏宽容的环境，日本人创业的成功率非常低，藤井说可能 1000 个创业者里只有 1 个成功。而且日本民族精神里讲究“和”，不愿意出头冒尖，更担心出格之后被孤立——日本社会往往把创业行为看作一种出格行为。

藤井还补充到，日本企业的“年功序列”也是日本人较少独立创业的原因。所谓“年功序列”，即国内所说的“排资论辈”，一些管理职位，没有在公司熬上多年的年轻人是够不着的；另一方面，年龄较长的人，即使没有很大的功劳，公司也还会轻易解雇，工资会随着工龄的增长而增长。在渡边纯一的《失乐园》小说中，男主人公和他的朋友就经历过多次“职级降低薪水不变”的情况，降级虽然会使人受挫，但日本人较少就此离开公司或创业。

不过藤井义彦也强调日本社会也在发生改变。就宏观方面来说，日本 1995 年经济泡沫之后，人们就开始认为要有创业精神，日本政府在政策方面也予以扶持。比较巧合的是，藤井义彦辞职发生在 1995 年——工作 30 年之后；自己所创办的公司，则是在经济泡沫三年之后。从这一点来看，他本人也是身体力行。

另一个变化就是“年功序列”制度的动摇。藤井说“年功序列”制度已经在 40 岁年龄段发生变化，现在大公司里 40 岁的日本人，有职位高的，也有职位低的，“在以前不是这样的”。日本年轻人对于“年功序列”开始存在不满，为什么工龄长就拿更多的工资，慢慢开始涌现更多创业。

不过即使有政府的支持，社会环境也在变得更宽容，日本年轻人走上创业道路的仍然不多，从日本创新企业在移动互联网潮流中集体失声就可见一斑。这可能要归究于经济环境。自 1990 年代经济泡沫以来，日本经济一直在执行通货紧缩政策，日本人实际上收入基本没有增加。时代变化非常快，生活压力也大，一些年轻的日本人就希望更稳定一些，这时候去大公司是比较好的选择。这个时代与藤井年轻的时候是相反的，他说那个时候经济整体向上，人们希望变化，希望求得更好的发展，“现在是经济下滑，大家求稳定”。藤井提到日本学生为了更稳定的生活，现在出国留学的学生也在逐年减少。

藤井义彦本人一直在倡导日本年轻人增强自己的主体意识，不应该是“灭私奉公”，而应“活私奉公”。他认为企业有自己的价值观，个体也有自己的价值观，一个人在公司工作必须要有个人理想，要有自己的主体性。不过他不认为日本人应该颠覆传统的“和”的团体意识，而是在现有民族意识上加上“主体性”意识。

这是一项长期的工程和使命，藤井会在多个场合演讲他的“活私奉公”倡议。不过他认为改变的关键还是在于日本人自己，“日本人总喜欢找借口，说‘失败不可原谅’‘社会环境不宽容’，不敢创业”，藤井说，“但年轻人要有创业的勇气，不要怕失败。日本人有更多自己的主体性，日本社会就会变得更好”。■

本文链接：
http://developer.51cto.com/art/201309/409437.htm

一个在清华附近蹲了17年的男人

注：在昨天国内互联网最大新闻就是腾讯入股搜狗了，那么你知道作为故事的主角王小川又是如何在 17 年里一步步走到现在的吗？

前十年

1996 年，四川省高中生王小川在国际奥林匹克信息学竞赛中获得金牌，被点招进了清华大学计算机系。站在北京五道口清华大学南门门口的 18 岁学霸，并没有传奇故事里想要闯出一番大事业的踌躇满志，也没有宅男穿越到新环境的恐惧，他面无表情——原因是，他对这一切的感觉很迟钝。

这种迟钝王小川自己心里也很清楚，他一直坦然接受着自己的性格，以及由性格所决定的命运。

1999 年，另一个更大的学霸：在美国麻省理工学院硕士毕业，后又拿到斯坦福大学商学院 MBA 的陈一舟回国，想要创办一家中国互联网公司。他来到清华，辗转找到几个“活好”的学生，王小川由此兼职进入 ChinaRen。一起进入的还有好几个清华计算机系 96 级的学生，包括后来连续创办点点网和啪啪的许朝军。

2000 年对王小川是个重要的年份，首先，ChinaRen 被当时如日中天的搜狐收购，张朝阳来了，陈一舟走了；其次，他大学毕业了。他做了一个换做今天可能 80% 的人都不会做的选择：把在搜狐的工作当兼职，继续在清华大学计算机系高性能所读研究生。

2003 年，研究生终于毕业了，实际上已经在搜狐干了 4 年、开发出了包括搜狐网站编辑后台的王小川同学正式加入搜狐，任高级技术经理。

2004 年，张朝阳对王小川说：我们来做搜索吧，王小川说好。搜狗成立，同年搜狐入驻清华科技园，他又回到了清华南门。隔着窗户就能看到清华，又招来好些学弟。

2005 年，一个好消息，一个坏消息。王小川升任公司副总裁，“扶了副”，可百度上市光辉璀

璨，搜狗很快迎来考验。这两年，好几个人从搜狐出来，投身了视频网站这个热门领域，比如古永锵和李善友；同学许朝军也离开，去了陈一舟收购的校内网（人人网）。

互联网靠 SP 业务挺过严冬，开始迎接有一个春天，热度越来越高。也有不在此具名的某些知名互联网公司和知名投资人找到王小川：要不要出来干？

公允来说，在搜狐已经五六年的王小川即便此时离职创业，不但算是个良机，道义上也无可指摘。可是，无关道德也无关商业，他终究是想都不想就留了下来。“其实我有时候是个迟钝的人，当别人看到机会兴奋时，我的兴奋点要迟一些。”王小川说，“而且我也不喜欢变化。”

他不喜欢变化，这是一种很复杂的情绪或本能，里面夹杂着对不可知未来的担心，或说害怕。“中学时我换了学校，换了环境，成绩一下就跌了下去。虽然最后也能再回前列，但要花好长一段时间。”

后七年

后面的故事大都已被人熟悉：2006 年搜狗输入法意外问世，并在之后两年横扫输入法市场，让王小川洞悉了客户端独特的产品和渠道玩法；2008 年在高压之下坚持开发浏览器，因为不被高层认可却固执坚持，一度被打入冷宫，“王小川擦擦桌子就有人以为他是要离职”，一年多见不到张朝阳；2010 年初，360 周鸿祎上门来谈入股搜狗，王小川一边施以“拖字诀”，一边跑到杭州找到马云入股，搜狗从搜狐分拆独立。

直到 2013 年 9 月 16 日，腾讯入股搜狗，将搜搜业务并入，以 4.48 亿美元获搜狗 36.5% 的股份。

十七年的价值

互联网大浪起伏，变化不断，有些公司起来了，如 360 重回搜索；有些公司消声了，如李善友的酷 6 网；有些人来了走了，陈一舟、许朝军、古永锵，乃至后来的龚宇，可王小川留了下来，还在搜狗，在清华科技园里。

在这 17 年里，他一直蹲在这里，一直蹲到马化腾和张朝阳配合地站在两边，他在中间腼腆微笑。

也许他本来会有更好的选择？或许如果之前搜狗真的没能做下去，会有一个世外高人打扮的老先生，用看透世事的眼神对他说：看，树挪死，人挪活呀！玲珑的人们，浮躁的社会，你总会有所谓更好的选择，在所谓的“聪明”里，隐藏的仍然是那一套成王败寇的故事。

在这样的环境下，王小川用自己的 17 年证明了一个常识：坚持做好一件事，终归有它的价值。■

原文链接：
http://developer.51cto.com/art/201309/410983.htm

C语言在2013年仍很重要

编者按

原作者注：在本文最开始，我并没说明进行模2^64的计算——我当然明白那些不是“正确的”斐波那契数列，其实我不是想分析大数，我只是想探寻编译器产生的代码和计算机体系结构而已。

本文作者在开发Dynym项目，这是一个动态语言的通用运行时。在开发时，作者以其 他语言的运行速度作为基础比较语言的运行速度，因此发现了一些小秘密。迭代计算斐波那契数列是测试各种语言执行速度的常见方法。作者以不同的语言进行测 试，最终发现C语言要比Python编写的计算斐波那契数列快278.5倍。在底层开发，以及专注性能的应用程序中，选择是显而易见的。而为什么会有如此 大的运行性能差距呢。作者进一步研究了程序的反汇编代码，发现差别出在内存的访问次数，以及预测的CPU指令的正确性方面。

原作者注：在本文最开始，我并没说明进行模2^64的计算——我当然明白那些不是“正确的”斐波那契数列，其实我不是想分析大数，我只是想探寻编译器产生的代码和计算机体系结构而已。

最近，我一直在开发Dynvm———一个通用的动态语言运行时。就像其他任何好的语言运行时项目一样，开发是由基准测试程 序驱动的。因此，我一直在用基准测试程序测试各种由不同语言编写的算法，以此对其典型的运行速度有一个感觉上的认识。一个经典的测试就是迭代计算斐波那契 数列。为简单起见，我以2^64为模，用两种语言编写实现了该算法。

用Python语言实现如下：

```
1. def fib(n):
2.     SZ = 2**64
3.     i = 0
4.     a, b = 1, 0
5.     while i < n:
6.         t = b
7.         b = (b+a) % SZ
8.         a = t
9.         i = i + 1
10.        return b
```

用C语言实现如下：

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. typedef unsigned long ulong;
4.
5. int main(int argc, char *argv[])
6. {
7.     ulong n = atoi(argv[1]);
8.     ulong a = 1;
9.     ulong b = 0;
10.        ulong t;
11.
12.        for(ulong i = 0; i < n; i++) {
```

```
13.             t = b;
14.             b = a+b;
15.             a = t;
16.         }
17.
18.         printf(“%lu\n”, b);
19.         return 0;
20.     }
```

用其他语言实现的代码示例，我已放在github上。

Dynvm包含一个基准测试程序框架，该框架可以允许在不同语言之间对比运行速度。在一台Intel i7-3840QM（调频到1.2 GHz)机器上，当n=1,000,000时，对比结果如下：

```
1. ==================================
2.  语言                    时间 (秒)
3. ==================================
4.  Java                    0.133
5.  C Language              0.006
6.  CPython                 0.534
7.  Javascript V8           0.284
```

很明显，C语言是这里的老大，但是java的结果有点误导性，因为大部分的时间是由JIT编译器启动（~120ms）占用的。当n=100,000,000时，结果变得更明朗：

```
1. ==================================
2.  语言                    时间（秒）
3. ==================================
4.  Java                    0.300
5.  C Language              0.172
6.  CPython                 47.909
7.  Javascript V8           24.179
```

在这里，我们探究下为什么C语言在2013年仍然很重要，以及为什么编程世界不会完全“跳槽”到Python或者V8/Node。有时你需要原始性能，但是动态语言仍在这方面艰难挣扎着，即使对以上很简单的例子而言。我个人相信这种情况会克服掉，通过几个项目我们能在这方面看到很大的希望：JVM、V8、PyPy、LuaJIT等等，但在2013年我们还没有到达“目的地”。

然而，我们无法回避这样的问题：为什么差距如此之大？在C语言和Python之间有278.5倍的性能差距！最不可思议的地方是，从语法角度讲，以上例子中的C语言和Python内部循环基本上一模一样。

为了找到问题的答案，我搬出了反汇编器，发现了以下现象：

```
1.  0000000000400480 <main>:
2.  247  400480:   48 83 ec 08        sub   $0x8,%rsp
3.  248  400484:   48 8b 7e 08        mov   0x8(%rsi),%rdi
4.  249  400488:   ba 0a 00 00 00     mov   $0xa,%edx
5.  250  40048d:   31 f6              xor   %esi,%esi
6.  251  40048f:   e8 cc ff ff ff     callq 400460 <strtol@plt>
7.  252  400494:   48 98              cltq
8.  253  400496:   31 d2              xor   %edx,%edx
9.  254  400498:   48 85 c0           test  %rax,%rax
10. 255  40049b:   74 26              je    4004c3 <main+0x43>
11. 256  40049d:   31 c9              xor   %ecx,%ecx
12. 257  40049f:   31 f6              xor   %esi,%esi
13. 258  4004a1:   bf 01 00 00 00     mov   $0x1,%edi
```

```
14. 259   4004a6:    eb 0e              jmp    4004b6 <main+0x36>
15. 260   4004a8:    0f 1f 84 00 00 00 00    nopl   0x0(%rax,%rax,1)
16. 261   4004af:    00
17. 262   4004b0:    48 89 f7           mov    %rsi,%rdi
18. 263   4004b3:    48 89 d6           mov    %rdx,%rsi
19. 264   4004b6:    48 83 c1 01        add    $0x1,%rcx
20. 265   4004ba:    48 8d 14 3e        lea    (%rsi,%rdi,1),%rdx
21. 266   4004be:    48 39 c8           cmp    %rcx,%rax
22. 267   4004c1:    77 ed              ja     4004b0 <main+0x30>
23. 268   4004c3:    be ac 06 40 00     mov    $0x4006ac,%esi
24. 269   4004c8:    bf 01 00 00 00     mov    $0x1,%edi
25. 270   4004cd:    31 c0              xor    %eax,%eax
26. 271   4004cf:    e8 9c ff ff ff     callq  400470 <__printf_chk@ plt>
27. 272   4004d4:    31 c0              xor    %eax,%eax
28. 273   4004d6:    48 83 c4 08        add    $0x8,%rsp
29. 274   4004da:    c3                 retq
30. 275   4004db:    90                 nop
```

（译注：

1、不同的编译器版本及不同的优化选项（-Ox）会产生不同的汇编代码。

2、反汇编方法：gcc -g -O2 test.c -o test;objdumb -d test>test.txt 读者可以自己尝试变换优化选项对比反汇编结果）

最主要的部分是计算下一个斐波那契数值的内部循环：

```
1. 262   4004b0:    48 89 f7           mov    %rsi,%rdi
2. 263   4004b3:    48 89 d6           mov    %rdx,%rsi
3. 264   4004b6:    48 83 c1 01        add    $0x1,%rcx
4. 265   4004ba:    48 8d 14 3e        lea
   (%rsi,%rdi,1),%rdx
5. 266   4004be:    48 39 c8           cmp    %rcx,%rax
6. 267   4004c1:    77 ed              ja     4004b0 <main+0x30>
```

变量在寄存器中的分配情况如下：

1. a: %rsi
2. b: %rdx
3. t: %rdi
4. i: %rcx
5. n: %rax

262和263行实现了变量交换，264行增加循环计数值，虽然看起来比较奇怪，265行实现了b=a+t。然后做一个简单地比较，最后一个跳转指令跳到循环开始出继续执行。

手动反编译以上代码，代码看起来是这样的：

```
1. loop:   t = a
2.         a = b
3.         i = i+1
4.         b = a+t
5.         if(n > i) goto loop
```

整个内部循环仅用六条X86-64汇编指令就实现了（很可能内部微指令个数也差不多。译者注：Intel X86-64处理器会把指令进一步翻译成微指令，所以CPU执行的实际指令数要比汇编指令多）。CPython解释模块对每一条高层的指令字节码至少需要六条甚至更多的指令来解释，相比而

言，C语言完胜。除此之外，还有一些其他更微妙的地方。

拉低现代处理器执行速度的一个主要原因是对于主存的访问。这个方面的影响十分可怕，在微处理器设计时，无数个工程时（engineering hours）都花费在找到有效地技术来“掩藏”访存延时。通用的策略包括：缓存、推测预取、load-store依赖性预测、乱序执行等等。这些方法确实在使机器更快方面起了很大作用，但是不可能完全不产生访存操作。

在上面的汇编代码中，从没访问过内存，实际上变量完全存储在CPU寄存器中。现在考虑CPython：所有东西都是堆上的对象，而且所有方法都是动态的。动态特性太普遍了，以至于我们没有办法知道，a+b执行integer_add(a, b)、string_concat(a, b)、还是用户自己定义的函数。这也就意味着很多时间花在了在运行时找出到底调用了哪个函数。动态JIT运行时会尝试在运行时获取这个信息，并动态产生x86代码，但是这并不总是非常直接的（我期待V8运行时会表现的更好，但奇怪的是它的速度只是Python的0.5倍）。因为CPython是一个纯粹的翻译器，在每个循环迭代时，很多时间花在了解决动态特性上，这就需要很多不必要的访存操作。

除了以上内存在搞鬼，还有其他因素。现代超标量乱序处理器核一次性可以取好几条指令到处理器中，并且“在最方便时”执行这些指令，也就是说：鉴于结果仍然是正确的，指令执行顺序可以任意。这些处理器也可以在同一个时钟周期并行执行多条指令，只要这些指令是不相关的。Intel Sandy Bridge CPU可以同时将168条指令重排序，并可以在一个周期中发射（即开始执行指令）至多6条指令，同时结束（即指令完成执行）至多4条指令！粗略地以上面斐波那契举例，Intel这个处理器可以大约把28（译者注：28*6=168）个内部循环重排序，并且几乎可以在每一个时钟周期完成一个循环！这听起来很霸气，但是像其他事一样，细节总是非常复杂的。

我们假定8条指令是不相关的，这样处理器就可以取得足够的指令来利用指令重排序带来的好处。对于包含分支指令的指令流进行重排序是非常复杂的，也就是对if-else和循环（译者注：if-else需要判断后跳转，所以必然包含分支指令）产生的汇编代码。典型的方法就是对于分支指令进行预测。CPU会动态的利用以前分支执行结果来猜测将来要执行的分支指令的执行结果，并且取得那些它“认为”将要执行的指令。

然而，这个推测有可能是不正确的，如果确实不正确，CPU就会进入复位模式（译者注：这里的复位不是指处理器reset，而是CPU流水线的复位），即丢弃已经取得的指令并且重新开始取指。这种复位操作有可能对性能产生很大影响。因此，对于分支指令的正确预测是另一个需要花费很多工程时的领域。

现在，不是所有分支指令都是一样的，有些可以很完美的预测，但是另一些几乎不可能进行预测。前面例子中的循环中的分支指令——就像反汇编代码中267行——是最容易预测的其中一种，这个分支指令会连续向后跳转100,000,000次。

以下是一个非常难预测的分支指令实例：

```
1. void main(void)
2. {
3.     for(int i = 0; i < 1000000; i++) {
4.         int n = random();
```

```
5.          if(n >= 0) {
6.              printf(“positive!\n”);
7.          } else {
8.              printf(“negative!\n”);
9.          }
10.         }
11.     }
```

如果random()是真正随机的（事实上在C语言中远非如此），那么对于if-else的预测还不如随便猜来的准确。幸运的是，大部分的分支指令没有这么顽皮，但是也有很少一部分和上面例子中的循环分支指令一样变态。

回到我们的例子上：C代码实现的斐波那契数列，只产生一个非常容易预测的分支指令。相反地，CPython代码就非常糟糕。首先，所有纯粹的翻译器有一个“分配”循环，就像下面的例子：

```
1. void exec_instruction(instruction_t *inst)
2. {
3.     switch(inst->opcode) {
4.         case ADD:    // do add
5.         case LOAD:   // do load
6.         case STORE:  // do store
7.         case GOTO:   // do goto
8.     }
9. }
```

编译器无论如何处理以上代码，至少有一个间接跳转指令是必须的，而这种间接跳转指令是比较难预测的。

接下来回忆一下，动态语言必须在运行时确定如“ADD指令的意思是什么”这样基本的问题，这当然会产生——你猜对了——更加变态的分支指令。

以上所有因素加起来，最后导致一个278.5倍的性能差距！现在，这当然是一个很简单的例子，但是其他的只会比这更变态。这个简单例子足以凸显低级静态语言（例如C语言）在现代软件中的重要地位。我当然不是2013年里C语言最大的粉丝，但是C语言仍然主导着低级控制领域及对性能要求高的应用程序领域。■

本文链接：
http://developer.51cto.com/art/201309/409226.htm

链 接

9月13日 今天是程序员节

程序员节是一个国际上被众多科技公司和软件企业承认的业内人士节日。日期是在每年的第256（十六进制为0x100，或28）天，也就是平年的9月13日或闰年的9月12日。

之所以选择256（28），是因为它是一个被程序员们所熟知的8位元基本数字。用1个字节（等于8位元）最多能表示256个数值，而且在平年中，256是2的最大幂中小于365的值。

白色被选为程序员节的主题色。在红绿蓝24位深（RGB）颜色空间里，每种原色可以有256种级别（从0到255一共256个数值）的深浅变化，当三种原色都达到最大深浅值，即为十六进制的0xFFFFFF时，表示白色。所以全世界的程序员们会穿着白色来庆祝这一天。

教你怎样玩转千万级别的数据

编者按

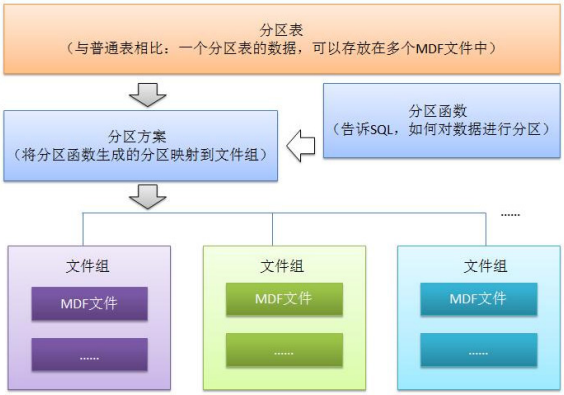
大数据处理是一个头疼的问题，特别当达不到专业DBA的技术水准时，对一些数据库方面的问题感到无奈。所以还是有必要了解一些数据库方面的技巧，当然，每个人都有自己的数据库方面的技巧，只是八仙过海，所用的武……

大数据处理是一个头疼的问题，特别当达不到专业DBA的技术水准时，对一些数据库方面的问题感到无奈。所以还是有必要了解一些数据库方面的技巧，当然，每个人都有自己的数据库方面的技巧，只是八仙过海，所用的武功不同而已。我把我最常用的几种方式总结来与大家分享，大家还有更多的数据库设计和优化的技巧，尽量的追加到评论中，有时一篇完整的博客评论比主题更为精彩。

方法1：采用表分区技术。

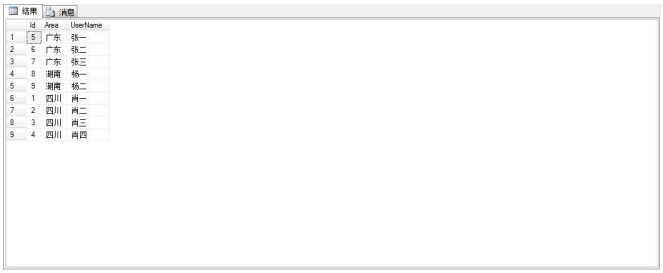
第一次听说表分区，是以前的一个oracle培训。oracle既然有表分区，就想到mssql是否有表的分区，当时我回家就google了一把，资料还是有的，在这我儿只是再作一次推广，让更多的人了解和运用这些技术。

表分区，就是将一个数据量比较大的表，用某种方法把数据从物理上分成若干个小表来存储，从逻辑来看还是一个大表。首先来个结构图：



上图虽然不能很清晰的表达表分区的执行过程，但是可以看出表分区要用到那些对象，比如数据文件，文件组，分区方案，分区函数等。

我们以一个用户表(TestUser)为例，假设这个表准备用来存储中国部分公民的数据，每条数据记录着每个人所属的省份(Area)，以及每个人的姓名(UserName)，如下图所示。当数据量达到1千万的时候，查询就比较慢了，这时候的数据优化就迫在眉睫。

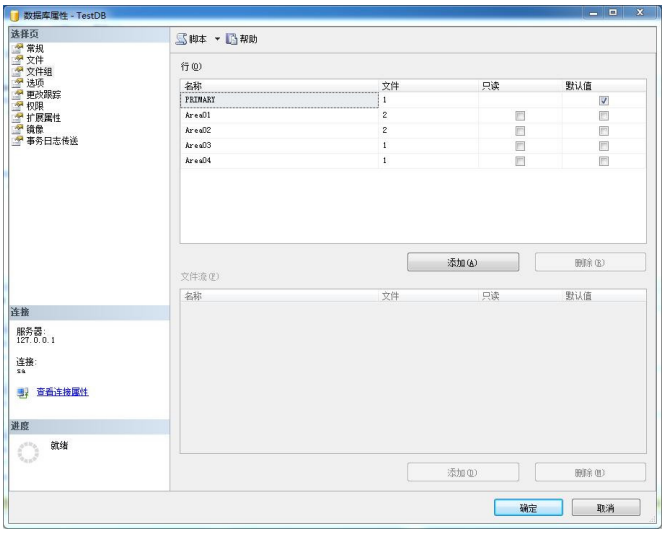


	Id	Area	UserName
1	5	广东	张一
2	6	广东	张二
3	7	广东	张三
4	8	湖南	杨一
5	9	湖南	杨二
6	1	四川	肖一
7	2	四川	肖二
8	3	四川	肖三
9	4	四川	肖四

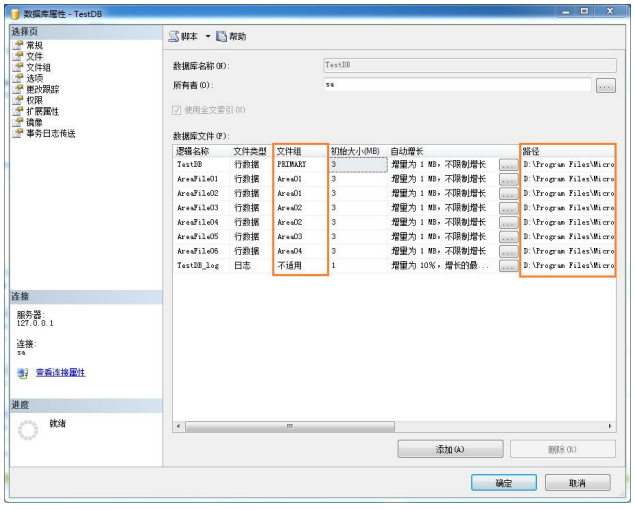
在优化之前，根据数据的结构，读写操作等，肯定会提出若干个解决方案。在这儿就以分区表的方案来优化数据库的查询，这儿以区域来分别存储数据，比如广东的公民存放在AreaFile01.MDF文件中，湖南的公民存放在AreaFile02.MDF的文件中，四川的公民存放在 AreaFile03.MDF的文件中，以此类推其它省份，为了实现这个功能我们就得做分区方案。在做分区方案时，首先要搞清楚分区方案要涉及到的四个对象：文件组，文件，分区函数，分区方案。

a：文件组，用来组织数据文件(.MDF)的一个虚

拟名称，一个文件组可以添加多个数据文件(.MDF)。打开SQL管理器，找到具体的数据库，然后右键【属性】，进入到【文件组】选项卡，添加Area01,Area02,Area03,Area04四个文件组。如图：



b：然后选择中【文件】选项卡，添加AreaFile01，AreaFile02，AreaFile03，AreaFile04，AreaFile05，AreaFile06六个数据文件（.MDF）,然后指定每个文件属于那个文件组(一个文件组可以存储多个数据文件)，以及这个文件的物理路径。在这儿大家已经看明白了，这些数据文件，就是物理上来分割一个数据表的数据的。也就是说一个表的数据有可能存储在AreaFile01中，也有可能存储在AreaFile02中，只要用某种方法来指定他们的存储规则就行了。



c：分区函数，就是指定数据的存储规则。就是告诉SQL，把新增的数据如何分区。创建一个分区函数，可以用下边的SQL语句来实现。

```
1. CREATE PARTITION FUNCTION
   partitionFunArea (nvarchar(50))
2. AS RANGE Left  FOR VALUES ( ‘广东’,’ 湖
   南’,’ 四川’ )
```

d：辛苦的创建了文件，又为其指定文件组，还建一个分区函数，目的只有一个，就是为了创建一个分区方案。分区方案可以用以下代码来创建。

```
1. CREATE PARTITION SCHEME
   partitionSchemeArea
2. AS PARTITION partitionFunArea
3. TO (
4.      Area01,
5.      Area02,
6.      Area03,
7.      Area04)
```

经过紧张的四步操作，一个分区方案就呈现在我们的眼前了。接下来的事，就是我们要怎样来消费这个分区方案。

首先我们创建一人普通的表，然后给这个表指定一个分区方案。如下代码。

```
1. CREATE TABLE TestUser(
2.      [Id] [int] IDENTITY(1,1) NOT NULL,
3.      [Area] nvarchar(50),
4.      [UserName] nvarchar(50)
5. ) ON partitionSchemeArea([Area])
```

为了能看到效果，再插入一些数据。

```
1. INSERT TestUser ([Area],[UserName])
```

```
Values(‘四川’,’肖一’);
2. INSERT TestUser ([Area],[UserName]) Values(‘四川’,’肖二’);
3. INSERT TestUser ([Area],[UserName]) Values(‘四川’,’肖三’);
4. INSERT TestUser ([Area],[UserName]) Values(‘四川’,’肖四’);
5. INSERT TestUser ([Area],[UserName])
   Values(‘广东’,’张一’);
6. INSERT TestUser ([Area],[UserName])
   Values(‘广东’,’张二’);
7. INSERT TestUser ([Area],[UserName]) Values(‘广东’,’张三’);
8. INSERT TestUser ([Area],[UserName])
   Values(‘湖南’,’杨一’);
9. INSERT TestUser ([Area],[UserName]) Values(‘湖南’,’杨二’);
```

查询所有的数据，可以用select * from TestUser;按分区查询：就用如下方法：

```
select $PARTITION.partitionFunArea([Area]) as 分区编号,count(id) as 记录数
1. from TestUser group by $PARTITION.
   partitionFunArea([Area])
2. select * from TestUser where $PARTITION.
   partitionFunArea([Area])=1
3. select * from TestUser where $PARTITION.
   partitionFunArea([Area])=2
4. select * from TestUser where $PARTITION.
   partitionFunArea([Area])=3
5. select * from TestUser where $PARTITION.
   partitionFunArea([Area])=4
```

效果图：

你们看我一个简单的表的分区是不是就已经完成了。呵呵，当然在实际应用中，仅仅掌握这点是不够的，比如在原分区方案上添加一个分区，删除一个分区。

方法2：用xml类型代替主从表设计，从而达到提高查询性能。

优化和提高数据库的性能，是从一个良好的数据库设计开始的。以一个会议预订系统为例，一个预订会议系统包括了会议时间，会议地点，主持人，参与人，知会人，记录者等相关信息。在的TDD，DDD模型主导的时代，在这儿为了更好的想表达我要阐述的问题，还是以表驱动模型来进行开发。

用户需求：

a：一个会议可能有多个主持人，虽然这种情况比较少，但是也有可能有。

b：一个会议有多个参与人，这个不难理解。

c：一个会议有可能要让某人知晓，这人可以参与或不参与会议，一般为高层。

d：一个会议有可能有零个或者多个记录者。

e：一个会议需要远程视频，投影仪，电脑，麦克风等会议设备中的某些设备。

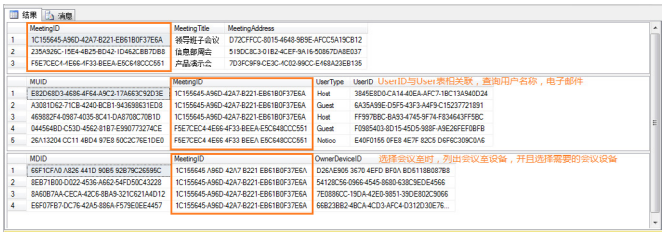
f：会议预订成功，或者会议时间，会议地点等重要信息修改后，邮件通知与会人员。

常规数据库设计：

a：建一个Meeting的主表，用于存放会议名称，会议地点，会议时间等的相关信息。

b：再建一个MeetingUser的表存储主持人，参与人，知会人，记录者。

c：同样，会议所需要的设备用MeetingDevice表来存储相关的信息。如图：



这样的表结构，是比较常规的设计方法，但是在实际应用中，你会发现一些待改进的问题。比如：

a：在提取一个会议的相关信息时，会连接多个表进行查询。这种查询在很大的程序上影响了数据库性能。

b：在做修改操作时也够呛的，先修改主表的相关信息，再把主表关联的子表信息全部删除重新插入一次，这样的操作是否够吐血了。当然有人精益求精，会比较修改前和修改后的数据，再用增加，删除，修改的手段达到子表数据的更新。这样的操作在有些ORM操作中已经实现了，但当自己code代码来实现的时候，特别是在多次code的时候，感觉总是那么烦心。

吐槽了这么多，是否有更好的解决方案呢？当然，在SQL里，我们可以XML数据类型来消除主从表的设计。如图：

上面的表结构设计，是不是有一个小清新的感觉呢？很明显，可以把第一种表的设计缺陷给消除了。一个会议的相关信息都存储在了一个表的一条记录中，这样的数据看起来是不是更直观呢？

a：获取一个预订会议的详细信息，我不需要进行多个表的连接查询，我要做的是只需用C#的Linq.Xml来解析查询出来的XML字符串即可。

b：修改操作时，我只需要重新组合XML数据，一个Update就更新了与会议相关的信息，操作是不是简单多了。

表面上看这种设计已经完美了，但是用户的需求是无止境的，有一天，你收到了一个需求，查询某个用户参与过的所有会议（就是只要主持人，参与人，或者记录者中包括了这个用户，就把这些记录都给查询出来），Oh！My God 这种表结构设计应该怎么解决这个问题呢？其实可以用XQuery解决这个问题，还没接触过XQuery的那得赶快充一下电了。

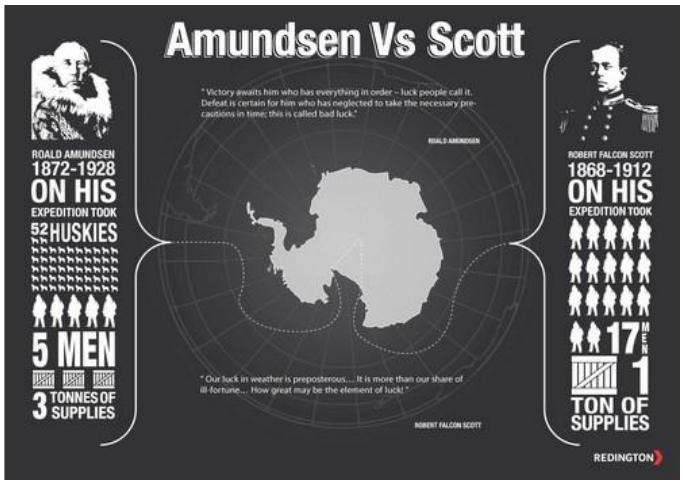
XQuery中最常用的有 exist(),value()这些函数，这儿就不详细的介绍了，网上搜索一下有很多相关资料，如果有必要，我会把以前项目中用的XQuery技巧与大家分享。■

本文链接：
http://developer.51cto.com/art/201309/410131.htm

从南极之争谈软件架构10个技巧及成功团队具备的气质

□ 廖煜嵘/译

随着云计算带来的低创业门槛、大数据潮流的盛行，越来越多的人加入了这场创业风暴。然而众多的淘金者中，真正满载而归的却是少之又少。这里为大家分享 HighScalabilty创始人Tod Hoff结合南极穿越之争带来的成功软件架构经验，及成功团队需具备的一些特性。以下为译文。



每个软件打造的核心都存在一次漫长的探险，或许你会觉得夸张，但是在 皇家卑诗省博物馆参观 Race to End of Earth（罗威探险家 Roald Amundsen和英国海军官员 Robert Scott于1911年–1912年完成的穿越南极之战）展览时，两支队伍采用的不同途径让我备受启发——那些同样存在于软件开发过程中决定成败原则。

我希望我可以重现 参观时的体验。随着参观的进行，我不断的对Scott的选择产生质疑，并叹服于Amundsen的老道，其中核心直指开发的两个极端Agile（Amundsen）与Waterfall （Scott）。

首先我们看两个比赛的背景知识

简而言之，率先抵达南极的队伍获胜。两个领队以完全不同的途径去完成这一目标，方法源于他们不同的目标、经验以及气质。比赛的结果是 Amundsen比Scott早33天抵达，同样队伍的状态上Amundsen更是远胜Scott。归途中，Amundsen队伍无损，而Scott小队 全军覆没。

更多的细节可以参考：

- Comparison of the Amundsen and Scott Expeditions
- 10 Mistakes That Caused the Most Punishing Nature Expedition in History
- The South Pole Fifty Years After

我们的重点则是可以运用于软件打造中的知识：

1. 单一的目标

这点让我感受颇深：Amundsen唯一的目标就是以最快的速度抵达目标，而Scott则要兼顾科学研究。

对于Scott来说，科学研究这个使命甚至优于资源与人员配备。而Amundsen的所有目的都是赢得比赛，还有平安归来。

想比之下Scott的双重目标存在很多矛盾。不错，那时确实不乏悠久的帆船科学研究历史（比如达尔文），然而达尔文并不是进行比赛，他们的船很大，同时科学研究只是附加目标，而环境也远没有南极那么恶劣及荒无人烟。

Fred Wilson在谈专注的力量时 经常引Steve Jobs为例——“我们专注于打造一个适合所有场景电脑的生产线，然后逐渐关闭其它的生产线”。专注是类似Minimal Viable Products及 time box scheduling产品背后的原动力，Amundsen专注在比赛上，所以一切策略都以这个为目标展开。

2. 使用简单，已被验证的技术

比赛中最难以接受的就是Scott的计划选择了一组复杂且未经验证的运输技术。

Amundsen的计划是简单的，他们选用狗拉雪橇；而在那时，狗拉雪橇的这个技术无疑是得到论证的，因此奇怪的应该是不选用狗的人。

然而在早先的旅行中Scott对狗有着非常不好的体验，所以他并未选用这个运输途径。取而代之的是，Scott选用了motor-sledge， 这是机动雪橇的早期版本。在那个时候，这还是个试验中的技术，最终3架motor-sledge都在途中损坏。然而当下的问题不是讨论他们为什么有那么差 的体验，而土著人民却非常善于用狗，而是缺乏对事物的详细了解并往下定论。

小马的任务是搬运补给，但是很小的蹄子让其不能胜任工作，因为它很容易受到潮湿和冰冻的侵害。9匹小马在旅行开始时就失去了作用，然而马和motor-sledge的补给只能储存在船上。对比而言，狗无疑更适应战场，它们可以吃南极洲捕获的企鹅和海豹肉。

使用人力拉沉重的雪橇，本来只作为应急，但是马和motor-sledge的缺失让这个方案执行了3/4的旅途。而雪橇还在不停变重，因为沿途他 们不得不收集一些岩石以作科学研究。这样，问题就在于他们根本没有足够的食物以支撑整个过程。他们并不清楚，一天吃4000卡路里的他们却需要消耗6000到10000卡路里的能量。

3. 定制、测试、重复

Amundsen的计划很周密，不留任何漏洞。当他发现设备达不到需求时，会自己动手，他亲自做了防风镜、滑雪板、犬绳及肉干。这种自力更生正是开发者需要具备的品质：

所有Amundsen的工具都出自自己的工坊，并经过一次又一次的提炼。Amundsen在制造工具时使用了两个信条：第一，远甚于批量生产的设备；第二，参与制造，可以确保设备在比赛中的表现。

4. 冷静且无情的

捕获焦点，必须具备认知重点的能力，只做必要的事情。

Walter Sullivan在The South Pole Fifty Years After使用另一个方式完美的阐述了这个道理：

登月通过一连串的火箭完成，而在这个过程中这些火箭被逐一抛弃；那个挪威人使用了同样的策略，在旅途中不断抛弃虚弱的动物，并且作为其它动物和人的食物。

5. 灵活

Amundsen原计划是去北极，但是在听说两个美国人已经抵达北极后，果断的将目标转向了南极，以获取世界第一这个奖励。

6. 从实际出发

Scott使用人拉雪橇不仅仅是为了运输，更加入了一种浪漫主义风格。Scott日记的背后甚至反射出Wagner的身影：

引用

在我的记忆中，没有任何与狗有关的探险可以达到这样的高度——人们直面险阻，并用自己的双手达到目的……无可否认的是，在这种情况下，征服才是更高贵、更华丽的胜利。

7. 技巧让一切都变得不同

Amundsen招募了多个经验老道的滑雪人并组成团队，相比，Scott的队伍无疑都是一些门外汉，并未针对需求进行训练。

8. 选择正确的团队

Scott的团队有许多来自英国的绅士，而Amundsen则选择了一些具有户外经验并有不同技能的工匠。

9. 错误的叠加

Scott的团队注定无法赢得比赛，然而他们的死亡却是由一系列的错误叠加引起。天气比预期的更冷，这样导致他们返程时在预定时间并未到达下一个 补给点，补给点存放了食物、燃料及其他物资。当Scott、Wilson和Bowers三人死亡时，离下一个补给点仅11英里远。马、motor- sledge及一个错误的补给点判断，让下一个补给点离他们遥遥无期。悲剧的铸成绝不因一个错误，而是由一系列错误叠加形成。

10. 后见之明

在事情开始时没有什么是清晰可见的，然而在结果产生时一切都已尘埃落定。Scott在开始时做了他认为最合适的决策，而其它人也同意他的观点。每 一个项目也是如此，没有人会愚蠢到从开始就放弃；然而通往成功的路径总是很少，并且沿途充满了太多的岔路。所以我们有必要去吸取一些成功的经验，多了解一 些最佳实践。

忆往昔，不难发现成功的团队总是具备一些共同的特性：规模小、良好的引导、专注、高技巧及具备丰富的经验，同时他们还有着健壮的计划、丰富的资源以及强大的战场适应能力。■

本文链接：
http://developer.51cto.com/art/201308/408250.htm

Java 与 .NET 的平台发展之争

■ Java 8即将正式发布，开发者Andrew表示，Java在某些特性上还是落后于.Net。比如，Java 8中最令人期待的Lambda表达式，在2007年发布的.Net 3.5中已经……

Java 8即将正式发布，从早期版本中，我们已经可以领略到一些令人兴奋的特性。但是开发者Andrew C. Oliver表示，尽管如此，Java语言在某些特性上还是落后于.Net。比如，Java 8中最令人期待的Lambda表达式，在2007年发布的.Net 3.5中已经存在了。他认为，.Net已有的和即将到来的特性要比Java 8优秀得多，如果Java 9再不做一些大的改进，那么Java落后于.Net就不止一点点了。

关于更新速率

微软有能力做出更快的改进。我记得在很早期的时候，微软能做到每周都更新数据库API：从ODBC、RDO、ADO到OLEDB等。自从出现了.Net之后，微软便达到了前所未有的更新速度。

但是Java为什么落后这么远？在早期的时候，Java的发展也是非常快速的，从Java 1.0.2 到 Java 1.1，仅仅一年时间，我们就看到了Java彻底地改变。从Java 1.1到Java 1.2只用了一年半时间，而Java 1.2.2只用了7个月的时间（这是一个重要的版本，只是使用了一个小版本号）。而在10个月之后，具有关键意义的Java 1.3问世，这也正是Java发行的第一个带有垃圾回收的版本。

Java 1.4为我们带来了NIO和正则表达式，但在之后不到两年的时间里就被取消了。Java 1.4.2版本带来了用于多核环境的垃圾回收器。Java 1.5带来了可用于生产环境的并行和并发GC（垃圾回收）特性，它还添加了更重要的并发和NIO功能，不过这一过程花了一年多的时间。

总的 来说，Java还是有不错的表现的，Java 6使锁变得更廉价，但其在本质上和Java 1.5是一样的，还是让用户多等了2年时间。Java 7是第一个对底层VM技术做出重大改变的版本，同时还给用户带来了invokedynamic特性——用于在JVM上更好地连接其它语言，但是在两个大版 本的更新之间用了大概5年时间，这个进度着实有些太慢了。

Sample features in Java and .Net and their release dates

Java feature	.Net or C# feature	Java version / release date	.Net or C# version / release date
java.util.concurrentFuture / ForkJoinPool / java.util.stream	Task Parallel Library	Java 5 / Sept. 30, 2004 Java 7 / July 28, 2011 Java 8 / April 2014?	.Net 4.0 / April 12, 2010
Lambda expressions	Lambda expressions	Java 8 / April 2014?	.Net 3.5 / Nov. 19, 2007
Strings in switch statements	C# switch	Java 7 / July 28, 2011	.Net 1.1 / April 24, 2003
Generics	Generics	Java 5 / Sept. 30, 2004	.Net 2.0 / Nov. 7, 2005
NIO	Asynchronous I/O	Java 1.4 / Feb. 6, 2002	.Net 2.0 / Nov. 7, 2005
Jigsaw	Assemblies and application domains	Java 9 / ?	.Net 1.1 / April 3, 2003 (subsequent incremental improvements)

为什么Java进展缓慢?

对于这个问题有一个简单的解释：Sun并不是一个实力超群的公司。Java创造于互联网繁荣时期，而那个时候Sun正在出售Sparc业务。

之后，互联网经济不景气，Sun决定持续加大其在硬件业务中的投入。Sun比较擅长创建生态系统，但它就是无法创造出用户需要的产品。Oracle是Sun的后继者，擅于彻底毁坏生态系统，最终吞并/摧毁圈内的同行，还会开发出高利润的产品来取代同行。

Oracle曾在一份简洁的公开声明中称：“我们都知道，由于各种商业和政治原因，该版本（Java 7）花费了不少时间。”

当 然，在分析Java的问题上，我们还必须考虑Sun公司的财政困难以及Java系统周边的东西。Sun公司违背了其提交Java进行标准化的初衷，它创造 了自己的“标准”委员会，即JCP（Java社区进程）。随着时间的推移，JCP尽管在一定程度上已经开放，但是无论是Sun还是现在的Oracle，都 拥有绝对的否决权，它们可以忽略规则，做任何想要的事情。

什 么阻碍了JCP? 不是开放性，而是利益冲突。我记得当时参与EJB3规范制定的某个供应商，它习惯延迟规范的进度。这是为什么呢？这些供应商需要购买或开 发一个产品来集成到它们的应用服务器中，如果下一代JavaEE规范已经发布，那么它们也必须尽快推出产品，它们不希望比市场晚。

协调产品的发布，对于一个公司来说都有些难，更不用说几个公司了。因此，我认为Java最大的问题并不是由于JCP造成的。

抛弃或分离一些东西

Sun已经成为了过去时，现在Oracle是“老板”，那么为什么Java版本的发布周期仍然需要这么长？最简单的解释是——Java太大。大项目往往意味着进展比较缓慢，且充满风险。下面我们就来看看如何将Java变得小一些。

首先，Oracle必须摆脱其“心爱”的客户端技术。当然，目前还没有更好的Swing和JavaFX的替代品，但是使用这些技术意味着需要把你捆绑在Oracle的平台上——至少目前是这样。

我 尚不清楚，目前JavaFX或客户端Java为Oracle带来的战略上的意义是什么，它们似乎被设计用来和VB6、Flash或一些4GL（第四代语 言）进行竞争的。在现代的、多平台的环境中，大部分人会认为触摸和滑动操作会更酷一些，而JavaFX与这种趋势是不相匹配的。为什么我们需要使用客户端 Java来阻碍服务器端的发展，并且还有可能伴随着各种风险，比如持续数月的Java零日漏洞安全问题以及关于如何禁用Java的讨论。

如今Java语言已经不再和Java平台一样重要。从Java平台中砍掉Java语言，并根据自己的时间表进行发布，这对于Oracle来说可能更容易——Oracle推出的开发工具不是Java业务的重要组成部分，并没有为大部分的Java开发者所使用。

Java 平台上有多种语言，比如JRuby、Scala等等。以高性能和可扩展的方式来支持这些语言和技术，对于云计算来说非常重要。如果云计算是未来，那么 Oracle应该首先考虑Java平台。而目前所支持Ruby、Scala、甚至Node.js的Java平台似乎是一个“锚”，而不是产生创新的“引 擎”。

比起Mark Reinhold（Java SE规范领导者，目前在Oracle公司），我更希望由Charles Nutter（JRuby创始人，目前在Red Hat公司）和Martin

Odersky（Scala创始人，目前在Typesafe公司）来决定在Java平台中添加哪些特性。我并没有不尊重Mark Reinhold的意思，但是一些证据表明，在很多与Java语言合作的项目中，Java语言拖慢了项目的进度。

对于Oracle领导的Java来说，事情发展不会那么顺利，很多Sun之前的决议现在仍然在困扰着我们。我的建议是，抛弃客户端Java，独立出JVM和Java语言的发布周期，致力于将Java作为一个平台，而不是想一次性地解决所有问题。■

本文链接：
http://developer.51cto.com/art/201308/408343.htm

专 题

一个神奇的网站：12306

9月11日是国庆假期抢票第一个高峰期，12306网站与抢票软件再次掀起风波。当日上午有大量使用抢票软件的网民反映无法登录12306网站，猎豹浏览器当晚在其官方微博声明，12306网站对使用了抢票软件的许多浏览器进行了技术屏蔽，且没给网民任何提示。据猎豹浏览器表示，目前已经解决了该问题，其用户可正常抢票。…>>详细

Java 6发现安全漏洞，建议尽快升级到7

安全研究人员呼吁使用Oracle Java 6的用户尽快升级到Java 7以避免成为活跃网络攻击者的受害者。

F-secure的高级分析师Timo Hirvonen这个周末通过Twitter发布了一个关于Java 6的安全警告，名为CVE-2013-2463.

PoC for CVE-2013-2463 was released last week, now it's exploited in the wild. No patch for JRE6... Uninstall or upgrade to JRE7 update 25.

— Timo Hirvonen (@TimoHirvonen) August 26, 2013

CVE-2013-2463问题Oracle已经在2013年6月发布的Java 7关键补丁更新中修复。但Java 6同样有此问题，但Java 6在2013年4月过后就不再更新，因此没有为Java 6准备的补丁。

云安全提供商Qualys对此漏洞的描述是：隐式的零日漏洞，而且可能被广泛利用。

而目前仍有大量的用户在使用Java 6，这一比例超过了50%！■

究竟什么是开发人员眼中最好的代码编辑器？

编者按

如果我们把不同的程序开发人员比作三国演义中的各路诸侯大将的话，那么代码编辑器绝对可以称之我们手中的神兵利器，不同类型的开发人员使用的”兵器“也大有 不同。

如果我们把不同的程序开发人员比作三国演义中的各路诸侯大将的话，那么代码编辑器绝对可以称之我们手中的神兵利器，不同类型的开发人员使用的”兵器“也大有 不同。好比兵器来说，没有绝对强的，也没有绝对好的，每一中兵器都有不同的优点和缺点，虽说俗话说的好，一寸长，一寸强，不过如果你没事去那都提着”关老爷“的“青龙偃月刀”得瑟，貌似也不是很方便。那么对于我们这些开发人员来说，究竟什么样的代码编辑器是最好的呢?

在今天的文章中，我们将从以下几个方面来比较各种类型的代码编辑器，评判指标包括：

- 友好度
- 功能性
- 扩展性
- 界面/体验
- 跨平台
- 价格

大师级别

vi

vi 对于使用过unix的朋友来说，绝对是再熟悉不过的代码编辑器，有多少伟大的程序和代码是由vi开发编辑的啊，既然是大师级别的代码编辑器，对于我们这些 普通人来说，只能说是好听不好用，基本上我周围的朋友使用vi的人大都是因为操作系统没有安装其它编辑器，也懒的花时间去安装。当然使用也相当麻烦了，你 得记住一大堆的命令，如果你记不住，唯一能做的就是关闭。我现在还记得我初次使用vi的时候，自己老老实实的打印了一张命令表，贴在墙上随时参考使用。

Vim

Vim 是一个类似于Vi的文本编辑器，不过在Vi的基础上增加了很多新的特性，Vim普遍被推崇为类Vi编辑器中最好用的一个。最早1991年发布，赢得了开源 世界的欢迎。和其它的代码编辑器相比不同的是命令行的工作方式。和简单的输入代码不同，你选择输入和选择文字，运行正则表达式的搜索，并且使用更多其它的 命令。vim使用脚本和插件可以变得非常适合扩展。可以支持GUI或者命令行。同时可以支持所有的操作系统。在大多数的Linux系统都预先装。

Emacs

Emacs 开发自1970，现在依然开发。这个编辑器拥有扩展，并且可以加载自定义的类库。它是第一个实现了代码高亮，自动缩进和多编程语言支持的代码编辑器。和 Vim一样，跨平台同时支持图形化界面和命令行。这个编辑器和LISP解析器整合，通过这个方式高手们可以修改它到极致。同时它是免费软件并且开源。

专业级别

Eclipse

Eclipse 是开发java应用的必备代码编辑器。这个IDE整合了插件结构，可以使得它轻松的支持其他编程语言。它拥有C/C++，Ruby，PHP和其它语言开 发。类似Google的功能开发自己版本的开发套件，所以可以很简单的创建Android和App引擎。免费并且开源。

Apatana Studio

Aptana 是一个专门为富客户端web应用开发设计的代码编辑器。基于Eclipse，并且帮定了强大的新工具。支持最流行的web开发语 言：PHP，javascript，HTML，css，Ruby，Python和其它更多插件。它拥有Git整合，能够部署你得应用到远程服务器。和 Eclipse一样，Aptana是免费和开源。

Netbeans

Netbean 是另外一个开发欢迎，和Eclipse一样，可以扩展支持其它的编程语言，PHP，Python，C/C++和其它。可以运行在 Linux，windows和OSX上。Netbeans可以快速的帮助你开发桌面应用，并且支持拖拽GUI，带来的负面影响就是性能差一些。但是这个 IDE免费并且开源

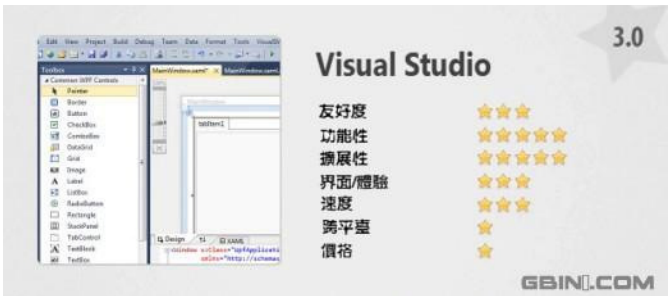
Dreamweaver

Dreamweaver 属于adobe应用套件之一，主要用来开发web应用。提供了最流行的web编程语言的支 持：PHP，ASP.Net，Javascript，HTML，CSS。主要为了初学者方便的编程，支持所见即所得的编辑方式。可以方便的部署到服务器，并且可以用来

开发jQuery移动应用。同时支持OSX和Window。单一价格$399。当然买套件更加合算。

Visual Studio

visual studio是一个All-in-one的windows开发环境。支持大量的开发语言(C/C++，C#，VB.NET和F#)。可以用来开发桌面应 用，移动和web。拥有强大自动补齐，行内文档，错误效验，debugging，表单设计，数据库schema设计。价格从$500开始，但是一个快速版 本的visual studio可以免费使用，我们可以使用有限的开发特性。



Xcode

Xcode 是一个Apple的解决方案，用来开发OSX和iOS应 用。支持C，C++，Objective-C，Objective- C++，Java，AppleScript，Python和Ruby。使用Xcode你可以书写，debug和预览代码。提供了GUI builder和一个移动设备模拟器用来测试iOS应用。IDE基于开源工具例如GNU Debugger和Apple LLVM compiler。Xcode曾经需要付费，但是现在免费提供给大家使用。

Coda 2

Coda是一个all-in-one的web开发人员工具。包含了FTP文件传输，代码导航，代码缩放，终端GIT整合，Mysql管理和其它。使用新的Coda2发布，你可以使用ipad作为一个预览屏幕。普通版本价格$99，但是你可以得到$75美元的折扣价。

设计级别

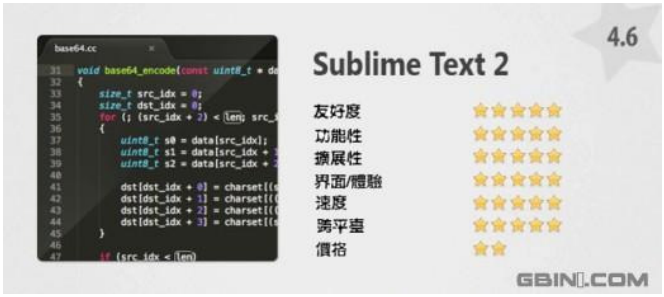
以下介绍的编辑器是轻量级，易于使用并且可扩展。这里有很多的支持社区，提供了插件，文章及其使用技巧。

Textmate

TextMate 是一个OSX上的常用图形文本编辑。包含了很多扩展的功能支持，包括：宏，bundle，代码缩放，代码片段，shell整合，剪贴板历史和项目管理。在 TextMate2中，我们期待带来很多需要的功能，比如，拆分视图和全屏视图支持。这个编辑器价格大概50美元。

Sublime Text 2

最近最火的编辑器非它莫属了，sublime 是一个超漂亮的跨平台编辑器。速度快并且功能丰富，几乎支持所有的编程语言。支持多行选择，代码缩放，键盘绑定，宏，拆分视图等等。同时拥有全屏和免打扰 模式。非常适合大屏幕的显示。和TextMate类似，拥有一个非常活跃的社区支持，而且开发了很多的插件和bundle，以前我们介绍过的使用sublime text 2开发Javacript和jQuery，我们可以看到Sublime的强大。它同时支持Linux，Windows和OSX。这个编辑器可以无限期试用。当然你可以花59美元购买，并且安装到任何一台你自己的电脑上。



普通级别

Notepad++

Notepad++是windows上的一个强大的轻量级编辑器。虽然名字好Notepad很像，但是功能更强大。支持几乎所有的编程语言，并且支持屏幕拆分，拥有FTP浏览器，宏及其强大的文本编辑功能。

一个免费的轻量级的OSX编辑器，支持多种编程语言。提供了强大的多文件搜索和替换功能，文字处理，文件比对，自动缩进，ftp等功能。

大家看到了，所有的编辑器sublime text 2的评分最高，随着最新版本的发布，随着跨平台特性，速度和使用的提升，sublime text 2将越来越受大家欢迎。

如果你有你自己最喜欢的代码编辑器，请给我们留言，我们将加到上面的列表中。 谢谢大家阅读！ ■

本文链接：
http://developer.51cto.com/art/201308/408273.htm

封面设计：@51CTO小林

《开发月刊》电子杂志

51CTO开发频道